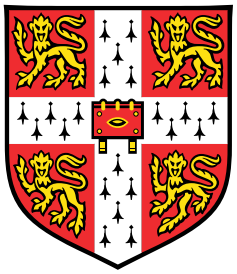
Semantic chunking



Ewa Muszyńska

Supervisor: Prof. Ann Copestake

Department of Computer Science and Technology
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy.

Clare Hall

October 2020

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. This dissertation is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as specified in the text. This dissertation does not exceed the regulation length of 60,000 words, including tables and footnotes.

Ewa Muszyńska
October 2020

Acknowledgements

I would like to express my deepest thanks to my PhD supervisor, Ann Copestake. Thank you for welcoming me into the department after my sudden change of course in the summer of 2015, and for patiently guiding me through both my Master's and PhD research projects. I could not have finished this thesis without your calm support and trust in my capabilities.

I am grateful to my fellow PhD students and members of the research group, in particular Alex Kuhnle, Guy Emerson and Matic Horvat, for shared learning, discussions and companionship.

My heartfelt appreciation goes to Andrew Kerr for his patience and support throughout the years of creating this thesis and for his contribution to spell checking the script. I would also like to thank my parents for their unceasing support.

Last but not least, my gratitude goes to the Engineering and Physical Sciences Research Council (EPSRC), who funded the research described in this thesis.

Summary

Long sentences pose a challenge for natural language processing (NLP) applications. They are associated with a complex information structure leading to increased requirements for processing resources. Although the issue is present in many areas of research, there is little uniformity in the solutions used by research communities dedicated to individual NLP applications. Different aspects of the problem are addressed by different tasks, such as sentence simplification or shallow chunking.

The main contribution of this thesis is the introduction of the task of semantic chunking as a general approach to reducing the cost of processing long sentences. The goal of semantic chunking is to find semantically contained fragments of a sentence representation that can be processed independently and recombined without loss of information. We anchor its principles in established concepts of semantic theory, in particular event and situation semantics. Most of the experiments in this thesis focus on semantic chunking defined on complex semantic representations in Dependency Minimal Recursion Semantics (DMRS), but we also demonstrate that the task can be performed on sentence strings. We present three chunking models: a) rule-based proof-of-concept DMRS chunking system; b) a semi-supervised sequence labelling neural model for surface semantic chunking; c) a system capable of finding semantic chunk boundaries based on the inherent structure of DMRS graphs, generalisable in the form of descriptive templates. We show how semantic chunking can be applied within a divide-and-conquer processing paradigm, using as an example the task of realization from DMRS. The application of semantic chunking yields noticeable efficiency gains without decreasing the quality of results.

Table of contents

Chapter 1

Introduction

The primary contribution of this thesis is a new task: **semantic chunking**. Long sentences pose problems for parsers, taggers and other systems commonly used in Natural Language Processing (NLP) because a large number of tokens usually indicates increased complexity of a sentence. The longest sentences are typically made up of multiple clauses, connected by various coordinating or subordinating relations. They often contain complex verb or noun phrases with enumerations, relative clauses, parentheticals, and so on, each introducing long-distance semantic dependencies between far-apart fragments. A successful NLP system needs to keep track of the dependencies to the same extent as humans do, but this proves to be a challenge incurring significant computational costs. For example, Figure 1.1 (McDonald and Nivre, 2011) shows the drop off in the accuracy of dependency parsing with the length of input for a graph-based and transition-based parsers.

Throughout this thesis, we will be using Dependency Minimal Recursion Semantics (DMRS; §2.3) as an example of a semantic representation. DMRS belongs to a family of flat compositional semantics referred to collectively as *MRS, and takes the form of a dependency graph. There exist several processors capable of yielding *MRS analyses, based on language-specific grammars.

Let us consider a sentence from a Wikipedia article:

(1) Marcellina has hired Bartolo as her counsel, since Figaro had once promised to marry her if he should default on a loan she had made to him, and she intends to enforce that promise.

Parsing Example 1 with one of *MRS tools, ACE (§2.4), produces 26,717 analyses before the processor reaches the default 1500 MB RAM limit. Apart from parsing, ACE is capable of performing the inverse task of realization, i.e. generating a surface string of a sentence from its semantic representation (§2.4.1). For ACE, the representation is *MRS.

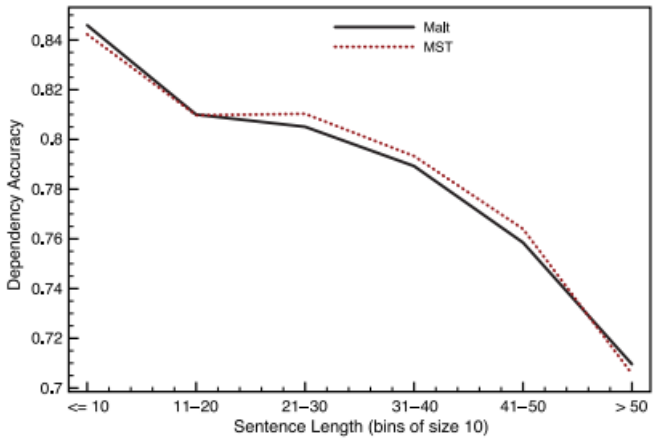


Fig. 1.1 Accuracy relative to sentence length. MST: graph-based; Malt: transition-based. Differences statistically significant ($p < 0.05$) at no positions. Figure reproduced from McDonald and Nivre (2011).

Realization is, however, not necessarily a one-to-one mapping. For instance, feeding the most appropriate *MRS parsed analysis of Example 1 back into ACE yields 5600 surface realizations.

The complex sentence can be subdivided into smaller constituents (§2.1.4). One choice results in the following:

(2) a. Marcellina has hired Bartolo as her counsel. (3 analyses, 2 surface realizations)
b. Figaro had once promised to marry her if he should default on a loan she had made to him. (12,036 analyses, 298 surface realizations)
c. She intends to enforce that promise. (3 analyses, 1 surface realizations)

The numbers in brackets show the results of a similar small experiment as for the full example. They indicate how many analyses are found for the sentence during the parsing stage and how many surface realizations the generator returns for the top variant. We can see that smaller inputs yield less ambiguous results.

The fragments above could have been produced by a task such as sentence simplification (§2.2.2). When considered together, they have the same meaning as the original sentence, since linking words, e.g. *and*, can mostly be deduced from context. The simplified set of sentences is coherent and we can imagine replacing the original sentence with the collection for improved readability.

Another choice of much finer constituents highlights boundaries between verb phrases:

(3) a. Marcellina has hired Bartolo as her counsel
b. Figaro had once promised

c. to marry her
d. he should default on a loan
e. she had made to him
f. she intends
g. to enforce that promise.

The resulting smaller fragments could be an output of shallow chunking of verb phrases (§2.2.1). These standalone pieces do not function well as independent utterances. Some of them also fail the parse-generation mini-experiment from earlier examples. For instance, the top analysis for Example 3c does not generate with a default ACE setup.

The two tasks we have brought up serve different purposes. Sentence simplification aims to improve the human experience, re-organising the syntactic structure or altering individual words if necessary. In the second example we isolated fragments commonly associated with shallow VP chunking, which has computational benefits. Shallow chunking is less costly than parsing, but it still provides some structural annotation for more complex downstream tasks. We discuss both sentence simplification and shallow chunking in more detail in §2.2.

This thesis presents a new general task devoted to lowering the computational cost of processing long sentences. We call it **semantic chunking**. The starting point of our investigation is the practical definition:

> Semantic chunks are semantically contained fragments of sentences which can be processed independently without loss of information.

The task evolved from existing and established tasks. Challenges posed by long and complex sentences plague many NLP applications, yet are often differently and individually addressed by their respective research communities. Semantic chunking constitutes an attempt to generalize across the field and provide a basis for a systematic treatment of the issue. The definition above highlights some distinguishing characteristics of the proposed task:

a) It is defined primarily by its applicability as a pre-processing step for more complex applications.

b) Semantic chunks are based on constituents that are self-contained with respect to the requirements of the target task and input representation.

c) The form of semantic chunks is flexible and depends on the intended application.

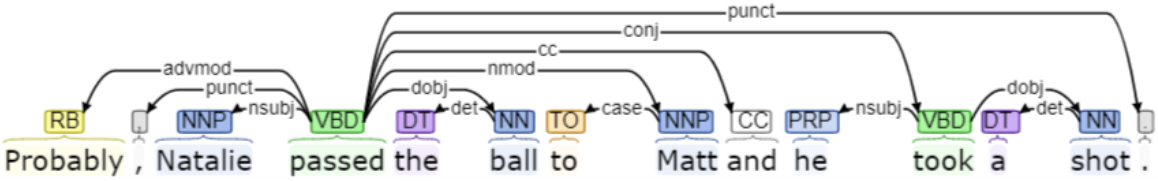


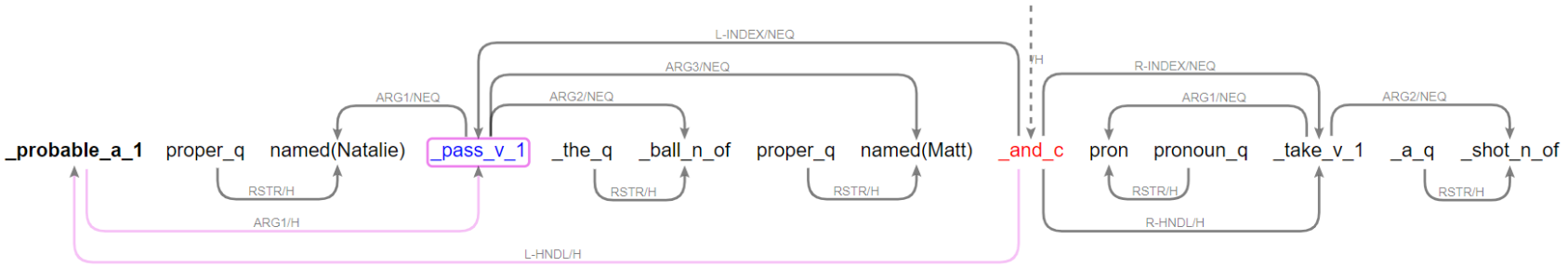
Fig. 1.2 Stanford dependency structure of Example 5. Parsed with https://corenlp.run/.

Semantic chunking can be applied to different representations. At this point, we would like to establish the term **sentence** to mean an utterance independent of its representation. The same sentence can be represented as a usual surface string, a sound wave if spoken, a syntax tree, or a dependency structure. The majority of the work described in this thesis focuses on complex semantic representations. We use Dependency Minimal Recursion Semantics (DMRS; §2.3) as an example, but the approach can be extended to other frameworks and representations. The focus on semantic, rather than syntactic constituents, makes chunking more relevant for higher-level tasks which care about text understanding, e.g. machine translation or summarisation.

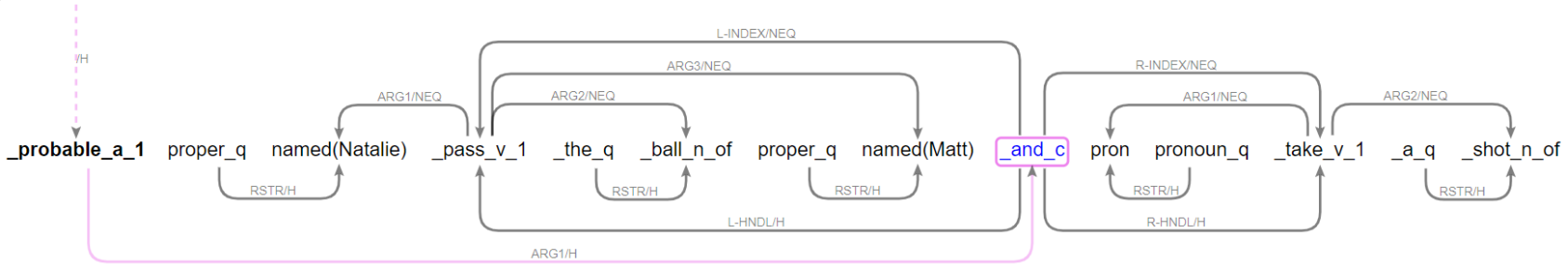
In Chapter 4 we address semantic chunking of the surface representation of sentences, i.e. their string. The semantic nature of the task means that it is difficult, if not impossible, to create an implementation based on regular expressions. For instance, let us consider the least ambiguous chunking scenario: clausal coordination. Two clauses connected by an *and* operator or just a comma are the most self-contained that any clauses combined in a single sentence can be. Although it would be tempting to create semantic chunks based on the presence of the coordinator string, such an approach would lead to false positives for every single non-clausal coordination in the dataset. As our experiments in Chapter 3 demonstrate, even a chunking system based on hand-written rules defined on a complex representation has limited capabilities.

Most semantic representations build upon syntactic analyses, so that there is a strong overlap between semantic and syntactic constituents. That is particularly true for DMRS, which is strongly compositional by design. Its semantics are anchored in the surface form of sentences and in syntactic links between constituents. Since semantic chunking relies on our ability to identify self-contained constituents in the chosen representation, simple composition principles of DMRS are particularly well-suited for exploring the new task.

Although important, syntactic relations themselves are not enough to fully determine semantic chunks. Additional information proves useful, for example, when considering the scope and nature of adverbial modifiers. Syntactic representations do not distinguish between those modifying a particular verb and modifying the entire clause.



(a) DMRS for Example 5 with *probably* scoping only over the left coordinate.



(b) DMRS for Example 5 with *probably* scoping over the entire sentence.

Fig. 1.3 Two scopal resolutions of Example 5.

(4) Quickly, Natalie passed the ball to Matt and he took a shot.

(5) Probably, Natalie passed the ball to Matt and he took a shot.

Quickly in Example 4 modifies the manner in which Natalie passed the ball, but *probably* from Example 5 has two possible interpretations dependent on its scope: either just the passing of the ball is uncertain, or the entire chain of events. Unlike syntactic dependency structures (Fig. 1.2), semantic representations, such as DMRS, are capable of distinguishing between the two scope resolution (Fig. 1.3a and 1.3b)[1]. The distinction is relevant for semantic chunking because in the latter interpretation the two coordinates are not independent and any chunking attempt has to preserve the information about the modifier being applied to both clauses.

The relationship between chunking and compositionality suggests a divide-and-conquer processing paradigm, where the full output of target tasks is built up from partial outputs based on individual chunks. This is supported by the practical requirement that chunks can be processed independently without any loss of information. Successful chunking will yield fragments such that the composed output will be indistinguishable from the result of processing the input in one piece. The approach benefits tasks which scale poorly with the size of the representation. For these, the composite approach outperforms the one-step analysis as the placement of chunking boundaries narrows down the search space of the solution. We demonstrate this result in our realization experiments (§§3.6 and 5.4).

Although semantic chunks aim to be self-contained, the information about their composition is part of the semantics of a sentence and has to be re-introduced when we construct the full analysis from the results for individual chunks. For instance, two potential semantic chunks for the DMRS in Figure 1.3a correspond to the two coordinates. The divide-and-conquer processing would apply the target task, e.g. realization, to each of them separately, yielding:

(6) Natalie probably passed the ball to Matt,

(7) he took a shot.

For the two string fragments to be correctly combined into a full sentence, we need to realize the linking coordinator *and* as well, and introduce it into the final result. The information about connections between the chunks, such the coordinator type, can be stored in auxiliary structures. We call them **functional fragments** and they are of the same type as the input representation, i.e. subgraphs for DMRS chunking or strings for surface chunking. The

[1] All DMRS figures in the thesis were produced with the Demophin tool and its `pydmrs` XML adaptation. For more details, see §2.4.

realization experiments of Chapters 3 and 5 propose some approaches to processing the functional fragments and how to use them for the construction of full analyses.

1.1 Outline of the thesis

Throughout this thesis we explore the task of semantic chunking, starting from its theoretical foundations. In Chapter 2 we connect the practical definition given above to theoretical concepts in semantics. We expand upon the intuitive requirements of semantic chunks, while homing in on a more rigorous theoretical foundation (§2.1). A key aspect of defining semantic chunks is finding balance between isolating self-contained constituents and remaining faithful to the computational considerations which motivate the task. Starting with primary units of meaning (§§2.1.1, 2.1.2), we address questions such as what form the information takes in a sentence (§2.1.3) and what makes a fragment of the representation self-contained (§2.1.4). The following section (§2.2) compares and contrasts semantic chunking with existing tasks.

The basics of semantic chunking are independent of the input representation, but for the purpose of this thesis, we explore the task for Dependency Minimal Recursion Semantics. Figure 1.3 displays examples of the representation. Its nodes correspond to individual predicates, while links express scopal and argument relations between the predicates. The remainder of Chapter 2 introduces the framework and relevant tools in more detail.

Chapter 3 describes our first attempts at implementing a semantic chunking system. These early experiments constitute a proof of concept and illustrate how the results of semantic chunking can be used in a processing pipeline. The prototype relies on a set of hand-designed rules (§3.2), which yield chunks in a highly restricted form of subject-verb finite clauses (§3.1). We also demonstrate how the results of semantic chunking can be used to improve the performance of chart generation with ACE (§3.6). The early experiments focus on the viability of the processing approach and, with that in mind, we prioritise the quality of semantic chunks over broad coverage, which should lead to fewer errors being propagated to downstream tasks.

Defining semantic chunking on a complex input representation benefits target tasks which already use the format as their input, as is the case for realization with ACE. The majority of NLP systems, however, tackle their objectives from the starting point of sentence strings and cannot be expected to rely on an often expensive annotation or parsing of a string into a more elaborate representation. At the same time, strings expose limited linguistic information without intermediate annotation or processing steps. In Chapter 4 we explore semantic chunking of sentence strings as a sequence labelling problem (§4.3), similar to the established tasks of shallow chunking and named entity recognition. The goal is to divide the

surface form of a sentence into fragments representing semantic chunks and any functional fragments required to preserve the information about how the chunks combine into a full sentence.

A common approach to sequence tagging tasks involves an annotated dataset, which is complex and expensive to create. We propose to bypass this cost by transferring chunks between representations to form a semi-supervised dataset. *MRS parsing maps predicates to spans of the source string. Thanks to this connection, DMRS chunks from Chapter 3 can be converted to surface semantic chunks (§4.1), which can in turn fuel a machine learning model, replacing costly human annotation. The semi-supervised framework fits the flexible nature of semantic chunking, allowing string-based models to be re-trained on a per-task basis.

Semantic chunks should take into account the requirements and limitations of their target tasks. Our experiments on surface chunks were performed with semantic parsing in mind. Parsing and realization are tightly coupled in the *MRS framework, and our design of DMRS chunks from Chapter 3 was guided by the dual objective of the inverse tasks. In order to evaluate the validity of our assumptions, we designed a DMRS-based F-score measure for intrinsic evaluation (§4.2) that allows us to rate how suitable the surface fragments we found are to act as semantic chunks for semantic parsing.

The intrinsic evaluation scores guide the creation of a dataset (§4.3.3) suitable for training a sequence labelling model (§4.3), based on the designs of the state-of-the-art approaches to similar sequence-labelling tasks. Our main architecture relies on a bidirectional long-short term memory (LSTM; Hochreiter and Schmidhuber (1997)) combined with a conditional random field (CRF; Lafferty et al. (2001)) decoder, following Ma and Hovy (2016) and Yang et al. (2018).

Figure 1.4 summarises the relationships between our realization and parsing experiments. The two inverse tasks relate the two representations we consider: DMRS and surface strings. Chunked DMRS graphs can be used for realization (top part of the figure), while chunked surface strings can be used for parsing (bottom). Additionally, semantic chunks can be transferred between the two representations as indicated in the diagram.

Successful experiments from Chapters 3 and 4 validate the task we propose in this thesis, but the rule-based proof-of-concept system laid out in Chapter 3 covers a narrow range of constructions and interprets the theoretical considerations of Chapter 2 in a minimal and rigid fashion. The hand-crafted approach stands at odds with the flexible design philosophy we advocate as the very premise behind semantic chunking. A task adjustable to the needs of its applications and input should not rely on expensive design effort, where a new set of

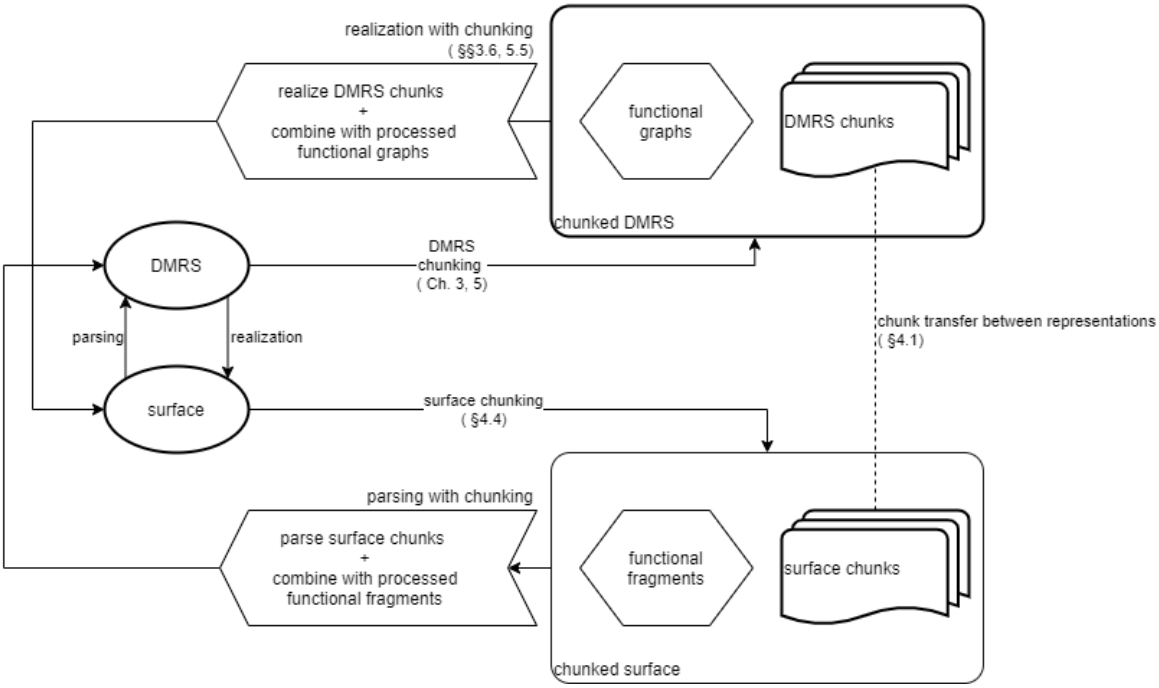


Fig. 1.4 Relationships between semantic chunking for parsing and for realization.

rules has to be implemented for every target task and representation, not to mention small variations due to framework variants or dataset idiosyncrasies.

In Chapter 5 we demonstrate how a tailored semantic chunker can be constructed automatically, as candidates for semantic chunks emerge from inherent properties of complex representations (§§5.1, 5.2). Their occurrences can be generalized into templates (§5.3). These repeating patterns are similar to the rules from Chapter 3 but bear the opposite relationship to the chunking process – they are the result of chunking the training dataset rather than its starting point. The templates allow us to filter the candidates for semantic chunks for criteria required by individual target tasks, and they express commonalities helpful in processing functional fragments of representations (§5.4.4). We illustrate these concepts by revisiting the task of realization from chunks (§5.4).

The greater flexibility granted by the departure from manual rules allowed us to experiment more freely with the shape of semantic chunks. The experiments presented in Chapter 5 apply the least restrictive criteria for valid DMRS semantic chunks for realization which still satisfy the theoretical guidelines of Chapter 2. While the chunks in Chapter 3 prioritized quality over coverage, the later experiments strongly target the latter.

Chapter 6 summarises the findings of the thesis and suggests further directions of research.

1.2 Contributions

Below we summarise the contributions of this thesis.

1. Introduction of a new task called semantic chunking.
    a) flexible goals, adaptable to a target application and input representation (cf. contribution 3);
    b) anchored in theoretical concepts of propositions and situations (Chapter 2),
    c) principles of chunking of the DMRS representation.
2. Three semantic chunking systems:
    a) A prototype rule-based chunker, creating DMRS chunks based on finite clauses (Chapter 3)
    b) A sequence labelling model which performs chunking directly on string representation (§4.3).
    c) A scope-based chunking system based on automatically detected chunks, generalized and tailored through the use of templates (Chapter 5).
3. Demonstration of how semantic chunking can be used in practice:
    a) Application of the rule-based and scope-based chunkers to realization (§§3.6, 5.4).
    b) Transfer of chunks between representations for efficient dataset creation (§4.1).
    c) Intrinsic evaluation and scoring for chunk quality (§4.2).
    d) Techniques for processing functional fragments (§§3.6.1, 5.4.4).

Chapter 2

Foundations of semantic chunking

In this chapter we explore theoretical foundations of semantic chunking, linking the intuitive understanding of properties required by computational aspects of the task to concepts in semantics. We explore what types of entities satisfy the empirical goals and constraints, and discuss concepts relevant to identifying fragments of sentence representations which can act as semantic chunks. After we consolidate our findings in §2.1.5, we give an overview of established tasks related to semantic chunking in §2.2, and then introduce Dependency Minimal Recursion Semantics (DMRS) and English Resource Grammar (§2.3), which form the main framework of our investigations.

2.1 Semantic chunking: an intuition

Semantic chunking is a practical task and the form of **semantic chunks** is flexible. Let us restate the definition or design specification which guides our investigation:

> Semantic chunks are semantically contained fragments of sentences which can be processed independently without loss of information.

Throughout the thesis we refer to **sentences** as representation-independent entities. The same sentence can be expressed in a surface string, a parse tree, or a complex semantic representation.

The goal of semantic chunking is to allow an operation on a full sentence to be replaced with a series of operations of the same type on smaller fragments, plus an additional reconstruction stage. If successfully applied, the combined procedure results in an outcome comparable with that of a single-step processing, but achieved with a lower computational cost. Since different target tasks focus on different aspects of the information contained in

their inputs, the performance and feasibility of our approach depends on the type of semantic chunks used. Their form has to take into account the requirements and limitations of the target task. The experiments described in §§3.6 and 5.4 demonstrate this processing paradigm for realization, while Chapter 4 explores chunking for string-based tasks, such as parsing.

Our discussion in this section sometimes invokes human interpretation of sentences, but semantic chunking is a computational operation, designed to improve the performance of machine processing, not processing by humans. This distinguishes it from some related tasks, such as sentence simplification (§2.2.2). The exact criteria of what constitutes a valid chunk should be tailored to the needs of the target downstream task and limitations of the input representation. The "lost information" from our practical definition is any information relevant for the target task which is not contained in the smaller fragments when compared to the full sentence. Its removal leads to differences between the reconstructed result of chunked processing and the result of processing the full sentence. The information which is inaccessible or irrelevant for the target task does not have to be preserved in semantic chunking.

As the name suggests, the focus of semantic chunking rests on assisting target tasks which care about the semantic content of sentences. The task could be adapted to the needs of processing longer discourse, given suitable representation, but that is outside the scope of our current investigation. The prioritized type of information is semantic meaning. We ignore pragmatic aspects of meaning and consider only the elements encoded in utterances removed from their wider context[1]. Furthermore, we focus on the text domain and do not address issues unique to speech. Although our experiments use English, the extension to other languages is possible, especially for representations like DMRS, which are supported by existing grammars of multiple languages.

2.1.1 Propositions

Semantic chunking focuses on preserving the semantic meaning of sentences. In this and the following sections we explore ways of formalizing this property as we identify relevant concepts in semantic theories. As the starting point, we take a popular and general notion of propositions. The guiding intuition behind the concept is that speakers uttering different declarative sentences can mean the same thing. All sentences in Example 8 convey the same information, which can be roughly expressed as the formula in 9:

(8) a. Bilbo gave Frodo Sting.

b. Bilbo gave Sting to Frodo.

[1] See e.g. Kearns (2011, Chapter 1) for the distinction between semantic and pragmatic aspects of meaning.

c. Bilbo dał Frodowi Żądło.

(9) `gave`(Bilbo, Frodo, Sting)

The predicate name in monospace font is just that – a name. For reference purposes it is the same as the English word which instantiates the predicate. In Example 8c the predicate is instead instantiated by the verb *dał* in Polish. Our work is limited to chunking English sentences, but the task is not language specific and can be adapted to other languages, provided availability of resources.

The two English variants of the sentence in Example 8 have different syntax. Their meaning is the same, but different word order gives particular elements of the sentence different importance and affects the accessibility of each constituent for discourse purposes. We do not attempt to preserve such information during semantic chunking.

The propositions in Example 8 consist only of a predicate and its arguments. The arguments are entities[2] bound in a relation expressed by the predicate, in this case Bilbo, Frodo and Sting, connected by the relationship of giving. According to the Stanford Encyclopedia of Philosophy (McGrath and Frank, 2018), the term **propositions** refers to some or all of the following:

- primary bearers of truth-value,
- objects of belief, doubt and other "propositional attitudes",
- referents of *that*-clauses,
- meanings of sentences.

Although the majority of philosophers seem to agree that the concept of propositions is required for fulfilling the roles listed above[3], there is no consensus as to the nature of such entities. Below we present a brief overview of popular theories, bur the origin of truth properties of propositions and their exact nature lie far outside the domain of this thesis.

One traditional way of discussing propositions is in the context of possible worlds. The possible worlds view associates linguistic expressions with extensions at various worlds, i.e. sets of entities that can be described by each expression in a given world. For declarative sentences, extensions are truth values. Intensions are functions from possible worlds to extensions (King, 2019). For each declarative sentence, its intension maps a given world to true if the sentence is true in that world, or equivalently, it is a set of worlds in which the

[2]The details of whether the participants in the relation are the entities themselves, their senses or some other aspects of their existence fall outside the scope of our discussion.

[3]See Chapter 2 of King et al. (2014), or McGrath and Frank (2018) for discussion.

sentence is true. This makes the intension the primary bearer of truth or falsity, which is one of the properties we ascribe to propositions. According to the possible worlds view, a proposition is equivalent to the set of possible worlds in which the proposition is true.

One argument against such a definition is that it leads to counterintuitive results. In particular, it does not account for the existence of distinct equivalent propositions. The two propositions expressed by sentences in Example 10 are true in the same set of worlds, i.e. all of them, but we are reluctant to assert that they mean the same thing.

(10) a. Sisters are female siblings.

b. A bachelor is an unmarried man.

The possible worlds definition of propositions is not compositional. Since our work focuses on linguistic representations of propositions, thinking of them as strictly truth-bearing objects is not useful. We consider the two propositions in Example 10 as having different meaning because they have different constituents. The compositional view of semantics holds that syntax and semantics are parallel, and for each phrase structure rule there exists a corresponding semantic rule (§2.1.4). According to this notion, propositions themselves are structured and contain the semantic values of words as their constituents. Such a treatment of propositions provides a natural way of accounting for the observation that sentences that are true in all the same worlds can express different propositions.

The classical theories of structured propositions descend from Frege and Russell. Both of them took propositions to be composed of the meanings of their constituents, reflecting how sentences which express them have grammatically structured constituents. They also asserted that propositions are platonic entities, independent of and originating outside of human cognition. While Frege's constituents were the Fregean senses (Sinne) of sentence constituents, Russell assumed the constituents to be individuals, properties and relations themselves. Soames and Salmon are responsible for what is probably the best known current theory of structured propositions (King, 2019). King et al. (2014) reject the platonic nature of propositions according to which propositions have truth values by their very nature and independently of minds and languages. They claim that a complete theory of propositions should explain how and why the propositions have truth values and represent the world.

In practical terms, propositions are typically expressed by declarative sentences. Syntactically, the simplest propositions correspond to subject-verb finite clauses. Such clauses can be used in a statement *I believe that X* in place of X. They can be true or false, even if the actual judgement is impossible without the context of the rest of the sentence or even knowledge of surrounding circumstances. Since finite clauses are the form taken by declarative sentences, most tasks will accept them as viable processing units, e.g. as root symbols in

parsing. Therefore, they are prime candidates for semantic chunks and the starting point of our practical investigation (§3.1).

At the same time, the choice of subject-verb finite clauses as semantic chunks is quite restrictive and leaves many complex sentences untouched, for example:

(11) After debating the course of action for a long time, the Council was surprised by Frodo, who offered to carry the Ring to Mordor.

The practical goals of semantic chunking require striking a balance between dividing the representation into smaller fragments that are easier to process than the whole sentence, but that are also larger than individual building blocks of the representation, whether those are words, predicates, or something else altogether. Reductive theories, as defined in Speaks (2014), describe propositions as special cases of other types of entities: properties, facts, and events. These different ways of thinking about propositions highlight aspects of sentence meaning that help to identify sentence fragments suitable to act as semantic chunks.

2.1.2 Situations and events

The notion of a proposition is too broad to directly assist a practical task like semantic chunking. We can link propositions to declarative sentences and finite clauses, but the concept gives us little guidance in regard to finer distinctions between suitable chunk boundaries. At the same time, propositions and the observation of their structured nature provide a starting point and foundation to investigate other ways of discussing sentence meanings.

An alternative way of thinking about the structure of propositions is in terms of situations. Situation semantics (Barwise and Perry, 1983) was developed directly as an alternative to possible world semantics (Kratzer (2019); cf. §2.1.1). It relies on the idea that utterances are about particular situations (Austin, 1950). A statement is true when the actual situation to which it refers is of the type described by the statement.

Barwise and Perry (1983) define situations as individuals having properties and standing in relations at various spatio-temporal locations. Individuals, properties, relations and locations are constituents, which can be put together in various ways to produce existing or hypothetical situations. The situations in turn are building blocks of the real world. Underspecifying location in a situation yields a situation-type – a function from relations and individuals to truth values. In our discussion, we typically refer to situations removed from context, which correspond to situation-types in Barwise and Perry's terminology. An utterance usually describes multiple situations, and it is sometimes useful to interpret a sentence as the class of such situations. For example, unqualified Example 12 can be

interpreted as describing any of the past situations when Sam cooked dinner, whether it was taters or a stew.

(12) Sam cooked dinner.

Situations bear similarities to events, the key entities of event semantics attributed to Davidson (1967). For example, the event in the statement *Sam cooked dinner* is a cooking of dinner by Sam. The introduction of events was motivated by the behaviour of verbs of action. The number and type of argument slots are generally thought of as part of the semantic content of the verb predicate. Under that assumption, every occurrence of the verb *cooked* in Examples 13 and 14 expresses a different predicate, since it takes different semantic arguments in each sentence.

(13) Sam cooked the fish.

(14) a. Sam cooked the fish on a bonfire.
b. Sam cooked the fish for dinner.
c. Sam cooked the fish on a bonfire for dinner.

The entailment between sentences suggests otherwise – all of the examples in 14 entail 13 but not vice versa. Event semantics accounts for this observation by adding one more argument place than usually recognised, filled by an event variable. For example, the Davidsonian analysis of the last sentence in Example 14 is:

(15) $\exists e[\texttt{cooking}(e,\ \text{Sam},\ \text{the fish})\&\texttt{for}(e,\ \text{dinner})\&\texttt{on}(e,\ \text{a bonfire})]$

Each predicate in 15 can be construed as a single proposition: a) Sam cooked the fish, b) the cooking was for dinner, c) the cooking was on a bonfire. Introduction of the Davidsonian variable adds a common element between these propositions and allows us to refine our notion of a semantic chunk, so that it encompasses all propositions sharing an event variable.

Parsons (1980) and others[4] took Davidson's logical representation a step further and introduced individual predicates for each argument. *Sam cooked the fish* would be represented as:

$$\exists e[\texttt{cooking}(e)\&\texttt{subject}(e,\mathit{Sam})\&\texttt{object}(e,\text{the fish})]$$

If we choose to follow the later developments, the first of the propositions in 15 can be further divided into: there was an event of cooking, the cooking was done by Sam, the cooking was performed on the fish. This level of granularity is too fine to meet the criteria required of semantic chunking. In most representations it would yield fragments corresponding to single token-level units.

[4]Further discussion in Casati and Varzi (1996), p. xiv.

The early Davidsonian account associated events only with action verbs. Actions are performed by actors, but there are events, such as *it rained*, where no particular subject can be identified. The line between whether something is an event or not shifts under the change of aspect. Does the sentence *it is raining* still refer to an event? Vendler (1967) established an ontology of verbs based on their temporal and aspectual types. The primary division was between events and states. Following a later categorisation in Moens and Steedman (1988), events are happenings with defined beginnings and ends, while states (of affairs) extend indefinitely. Events can be further divided into atomic and extended, based on their duration and relationship to other events. Bach (1981) coined the term *eventuality* to conveniently encompass states (Example 16), processes (Example 17) and events (Example 18).

(16) Hermione knows the potion recipe.

(17) Luna was writing an article.

(18) Neville dropped a book.

The neo-Davidsonian paradigm assumes that any predicate may have a Davidsonian event argument – not only verbs, but also adjectives and prepositions. We discuss the suitability of fragments centred around various parts-of-speech for the role of semantic chunks in the next section. At this point, focusing on the general concept of events, it is useful to think back to situation semantics. Higginbotham (1985) suggests that "the bundle of objects with event arguments might well be called [Barwise and Perry's] 'situations'". The goal of bringing together situation semantics and Davidsonian event semantics, at least in certain areas, is pursued in a number of works (Kratzer, 2019).

A key distinction between events and situations is that the event in Example 12 is not an event *in which* Sam cooked dinner. Sam cooking dinner is all there is to the event. In contrast, situations are not exclusive. We can define a partial relation, $s \leq s'$, which holds true if everything that occurs in the situation s occurs in s' but not the other way around. For example, the situation *Sam cooking fish* is part of the situation *Sam cooking fish for dinner*. Kratzer (2002) leveraged this relation to link situations and propositions:

> A situation s exemplifies a proposition p iff for all s' s.t. $s' \leq s$ where p is false, there exists s'' s.t. $s' \leq s'' \leq s$ which is a minimal situation where p is true.

Let us unpack the above statement, using the example from Kratzer (2002). The proposition p states that there are teapots. A minimal situation s'' where p is true comprises of two teapots and nothing else (Fig. 2.1c). If we remove a single teapot, p is no longer true. Let us consider two situations: s_1 with three teapots (Fig. 2.1a) and s_2 with three teapots and a pair of scissors (Fig. 2.1b). The situations labelled s' in the definition are parts of s_1 or s_2

(a) Situation s_1: three teapots.

(b) Situation s_2: three teapots and a pair of scissors.

(c) Situation s'': two teapots.

Fig. 2.1 Situation related by partiality relationships (from Kratzer (2002))

.

for which p is false. For s_1 these are situations s'_1 with a single teapot or no teapots. The set of s'_2 situations additionally includes all those situations enhanced by the pair of scissors. Scissors are completely absent from the minimal situation s'' and for any s'_2 with scissors $s'_2 \not\leq s''$. Hence, the situation s_2 does not exemplify p. On the other hand, situations with zero or one teapots (s'_1) all satisfy $s'_1 \leq s''$, so that s_1 exemplifies p.

Notions of partiality and exemplification bridge situation and event semantics. Davidsonian events can be thought of as minimal situations exemplifying the relevant propositions. The eventuality in Example 13 can be construed as a situation that exemplifies the proposition *Sam cooked*. We could represent the proposition, ignoring tense, as:

$$\lambda s.\exists x[\texttt{fish}(x)(s)\&\exists e[e \leq_p s\&\texttt{cook}(Sam)(x)(e)]]$$

The event variable e introduces a situation which is a part of the main situation described by the sentence, s, i.e. $e \leq_p s$.

Association between semantic chunks and situations suggests the following chunking of Example 11:

(19) After [debating the course of action for a long time]$_1$, [the Council was surprised by Frodo]$_2$, who [offered to carry the Ring to Mordor]$_3$.

We discuss the suitability of these fragments as semantic chunks in later sections, together with the role played by linking words, such as *after* and *who* in the example above.

2.1.3 Syntactic form of semantic chunks

The neo-Davidsonian analysis of utterances in terms of eventualities divides sentences into fine-grained propositions associated with individual predicates or even predicate arguments. The generalisation of eventuality arguments to other parts of speech leads to undesirable

proliferation of semantic chunks in sentences, if we insist on strict association between chunks and situations. Below we discuss several scenarios where a more careful consideration is needed.

Let us consider Examples 20 and 21.

(20) The fall of Barad-dur marked the beginning of a new era.

(21) When Barad-dur fell, a new era began.

The fall of Barad-dur expresses the same situation as the statement *Barad-dur fell*, just as *the beginning of a new era* means that *a new era began*. The noun phrases are nominalizations of the respective finite clauses. We have discussed in the preceding section why limiting semantic chunks to finite clauses is too restrictive. Treating every noun phrase as a potential chunk misses the mark in the opposite direction. For example, in English, deverbal nouns formed with suffixes like *-ing*, *-ation*, *-ment*, and *-al* are ambiguous between readings that can broadly be characterized as eventive and non-eventive (Lieber (2018) for an overview). The dinner in Example 22 refers to the food prepared for eating, while the dinner in Examples 23 is the event of dining as part of a situation involving people meeting up.

(22) The dinner took three hours to prepare.

(23) The dinner lasted three hours and everyone had fun.

Vendler (1967) conceived a linguistic test for distinguishing between different types of nominals. He suggested dividing them into perfect nominals, in which the process of nominalization is complete, and imperfect nominals, where the verb is still 'alive and kicking'. The distinction was based on what syntactic constructions accept the phrase. The two kinds of nominals also allow different types of modifiers, e.g. adjectives for perfect nominals (24) and adverbs for the imperfect ones (25).

(24) Hermione ruined the potion with the accidental addition of cat hair.

(25) Hermione ruined the potion by accidentally adding cat hair.

Chomsky (1970) names three types of nominalization: derived (Example 24), gerundive (25) and mixed. The "object" of gerundive nominals is a noun phrase rather than a prepositional phrase and modifiers are adverbial, while mixed nominalization (Example 26) accounts for phrases with the internal syntax of noun phrases rather than sentences, requiring a post-nominal prepositional phrase and adjectival modification:

(26) the accidental adding of cat hair

For the purpose of semantic chunking, derived nominals in English are difficult to reliably distinguish from non-eventive noun phrases. We do not consider them suitable candidates for semantic chunks and focus solely on *ing* nominals.

Gerundive nominals are indisputably verbal in nature and allow only for the eventive reading, but mixed nominals behave as noun phrases. They occupy argument slots commonly filled by nouns and can appear in coordination with non-eventive noun phrases. Coordination syntactically connects two constituents of the same kind. Consider Example 27:

(27) Sam's singing and a bonfire made Frodo's day better.

If we allow semantic chunks based on mixed nominals, *Sam's singing*, identified as a self-contained semantic constituent corresponding to a situation, has the same status as a simple object (*a bonfire*).

If we disqualify all noun-like constructions, we end up with the interpretation of semantic chunks as related to syntactic clauses. They are typically centred around a main verb, but, importantly, the verb does not have to be the main predicate of the situation. Copular verbs, such as *was* in Example 28, do not clearly contribute meaning to the sentence. They appear in utterances of the form *A is B*, where *B* is often a predicate of *A*, attributing a property. The adjective *angry* in Example 28 is predicative and the copula only provides support for the tense of the clause.

Our analysis assigns two semantic chunks (bracketed) both Examples 28 and 29:

(28) [Gollum was angry]$_1$ because [Sam cooked the fish]$_2$.

(29) [Gollum threw a tantrum]$_1$ because [Sam cooked the fish]$_2$.

Although the main carrier of meaning in the main clause of the first sentence is an adjective (*angry*), the main clauses of both sentences describe parallel situations, resulting from Sam's actions but distinct from the situation in which Sam cooked the fish.

For comparison, the adjective *angry* in Example 30 appears in an attributive role.

(30) Angry Gollum threw a tantrum.

Predicates used in attributive context are not valid bases for distinct semantic chunks, even if the same predicates qualify when used as primary predicates in the clause. Although they can be interpreted as eventualities in the neo-Davidsonian sense, attributive adjectives do not support complex modification (Examples 31 and 32) and are associated with short phrases, often corresponding to single tokens in the string representation.

(31) The hobbit was stupid enough to ruin the meal and he cooked the fish.

(32) *The stupid enough to ruin the meal hobbit cooked the fish.

The example of copular verbs illustrates the recurring issue of **granularity** and the search for the 'Goldilocks zone'[5] of chunking (§2.1.1). A semantic chunk corresponding to a single surface token or a single unit in any type of representation does not serve well the goal of reducing the complexity of a sentence.

The issue is further exacerbated by the existence of verbless clauses (Huddleston and Pullum, 2002, Chapter 14):

(33) Her face pale with anger, she left the tent.

(34) Her face reddening with anger, she left the tent.

Even though only the second sentence contains two verbs (*left* and *reddening*), the first one can be interpreted as equivalent to:

(35) Her face was pale with anger as she left the tent.

The new sentence (Example 35) supports two situations, which suggests that the same is true of the original utterance (Example 33). As a consequence, *her face pale with anger* can be construed as a valid semantic chunk. A qualification is, however, necessary: a valid chunk under the right circumstances. Whether a fragment, such as a verbless clause, can be treated as a semantic chunk depends not on whether we, humans, can identify it as referring to a specific, separate situation, but whether the chunking algorithm can recognise it reliably as such in the input representation and whether the target task can process it correctly without full access to the remainder of the sentence. We discuss these questions for the particular case of DMRS representation throughout the thesis.

2.1.4 Relationships between chunks

The premise of semantic chunking is based on the assumption that a sentence comprises smaller meaningful constituents. The same assumption lies at the heart of the principle of compositionality, which can be stated as

> The meaning of a complex expression is determined by its structure and the meanings of its constituents. (Szabó, 2017)

For every application of a syntactic rule, a semantic rule is applied which combines the meanings of the participants of the syntactic relation. The principle is commonly considered necessary for explaining the productivity and systematicity of language.

[5]The term originates in the fairy tale *Goldilocks and the Three Bears*. In astronomy, it describes the habitable zone around a star, where the conditions are just right – not too hot, not too cold.

There are several ways of identifying constituents, all of them testing the extent to which a fragment behaves as a unit. For example, constituents can be replaced by a constituent of the same type without altering the grammaticality of the sentence, even if it might change its meaning or render it nonsensical. They can be moved around in the sentence, omitted or inserted as a whole. We have already invoked such a test when we disqualified mixed gerunds as candidates for semantic chunks in §2.1.3, based on their behaviour as noun phrases.

Constituents are defined syntactically, but at the same time, they are coherent units of meaning. They isolate entities described by the sentence: events, individuals, properties. Their sizes range from words through phrases to clauses, and the smaller constituents are composed hierarchically to form larger ones. The syntactic hierarchy translates into a hierarchy of meaning constituents, and different representations highlight it to a different extent. The constituents are made explicit by the subtrees of a parse tree, but are less visible in dependency graphs or surface strings. Despite that, even in less explicit representations, constituents behave as units. In surface strings they are usually represented by consecutive text spans. When constituents are nested within one another, there exist practical limits on the depth and length of the nested clauses (De Vries et al., 2011). Dependency graphs have fewer edges going between constituents compared to the number of edges criss-crossing a single constituent. The semantic chunking techniques we propose in this thesis leverage such underlying patterns and structural properties of representations to identify suitable chunks (§§3.2, 5.1).

Let us go back to Example 11, repeated with marked potential chunks as:

(36) After [debating the course of action for a long time]$_1$, [the Council was surprised by Frodo]$_2$, who [offered to carry the Ring to Mordor]$_3$.

We will borrow terminology from Rhetorical Structure Theory (RST, Mann and Thompson (1988)), which is a framework for describing the coherence of text in terms of relations between its fragments. An important type of relation is that between a nucleus and a satellite, in which the satellite provides additional information or context about the nucleus. We will refer to a complex sentence as having a nucleus chunk and satellite chunks, but we adapt the terminology and our usage does not correspond strictly to the RST definitions. The units participating in relations outlined in RST are, however, similar to our candidates for semantic chunks. The fragments are essentially clauses, except that clausal subjects and complements, and restrictive relative clauses are considered as parts of their host clause units rather than as separate units. The constraints arise from the requirement that the units have independent functional integrity, which is similar to the requirement of chunks as self-contained fragments. Whether or not the special cases excluded by RST qualify as semantic chunks depends on the representation and target task in question.

The nucleus-satellite structure can be inferred from the interpretation of semantic chunks as individual situations. In §2.1.2 we discussed the connection between situations and eventualities in terms of a partiality relation. The nucleus chunk usually describes the main situation and satellite chunks describe supplementary situations that provide additional information. They narrow down the subset of situations exemplifying the proposition. They describe in more detail the entities participating in the situation, and provide explanation or wider context to the main situation. For instance, the nucleus of Example 36 is the second chunk (*the Council was surprised by Frodo*), with the other chunks acting as its satellites.

The key question for semantic chunking is whether satellite clauses have sufficient autonomy to satisfy the practical requirements of chunks. Mann and Thompson (1988) notice that satellites are incomprehensible without the nucleus but not vice versa. They can be often replaced by a different satellite fulfilling the same role without much change to the apparent function of the text as a whole, while the replacement of the nucleus is more drastic. That is true for a human reader, but does not necessarily hold from the computational perspective. We know that it is the Council that did the debating and Frodo who did the offering, yet if we look just at surface strings of the chunks, the actors of the satellite chunks are not clear. On the other hand, a potential simplified logical representation of the example can look something like:

$$\begin{aligned}\lambda s.[\exists e_1.[&\texttt{surprise}(e_1, \text{Frodo, the Council})\\ &\&\,\exists e_2, e_3[\texttt{offer}(e_2, \text{Frodo}, e_3)\,\&\,\texttt{carry}(e_3, \text{Frodo, the Ring})\,\&\,\texttt{to}(e_3, \text{Mordor})]\\ &\&\,\exists e_4[\texttt{debate}(e_4, \text{the Council, the course of action})\,\&\,\texttt{a long time}(e_4)]\\ &\&\,\texttt{after}(e_1, e_4)]\end{aligned} \quad (2.1)$$

Each line in the formula corresponds to a potential semantic chunk for some particular task. The chunk concerning Frodo's offer could be split further into chunks based on the offering situation and carrying situations, depending on the chosen level of granularity. Even though the actors of the satellite chunks are not explicit in the surface representation, the logical representation can express them directly by populating the argument slots of predicates `offer` and `debate`. Semantic chunks defined on such a representation preserve the information about actors so that it can be accessed by the target task if relevant.

Some predicates, like `after` in Example 36, do not belong to any semantic chunks. Instead, they act as connectors defining relationships between chunks. Functional expressions of this kind do not have independent meaning but they modify and combine expressions with semantic content. In syntactic analyses they do not form constituents of their own,

but instead they head phrases made up of themselves and their operands. For example, a preposition *into* is not a valid constituent on its own. It is typically analysed together with its argument as a prepositional phrase, e.g. *into Mordor*. We will refer to these elements as **functional fragments**. Semantic chunking aims to facilitate processing of long sentences by allowing the task to deal with individual chunks one at a time and then to combine the results. Additionally, the processing has to account for the functional fragments and their contribution to the meaning of sentences. In §§3.6 and 5.4.4 we suggest some ways of addressing the issue.

2.1.5 Semantic chunking: insights so far

In the preceding sections of this chapter we linked the informal definition of semantic chunking to established concepts in theoretical semantics. To reiterate, the basic premise of the task is that:

> Semantic chunks are semantically contained fragments of sentences which can be processed independently without loss of information.

We established that the general type of information which semantic chunking aims to preserve is semantic meaning. The extent to which this objective is pursued depends on the nature of the target downstream task meant to benefit from the pre-processing step. If a particular task or representation cannot access certain aspects of sentence meaning, there is no obligation for semantic chunking to preserve it. Although the definition could be adapted to include discourse elements, this is outside the scope of our investigation.

Propositions are commonly considered a basic unit of meaning, but we found them to be too vague and too broad a notion to offer direct benefits to a practically focused application. Their structured nature, however, lends itself to reconceptualization in terms of situations and event semantics (§2.1.2). The concept of situations, in particular, matches well the constraints of semantic chunks. Not only does it segment a sentence into semantically contained fragments, but provides a framework in which we can discuss relationships between individual semantic chunks in the sentence. For this purpose we borrow terminology from Rhetoric Structure Theory (RST) and organise situations and their associated semantic chunks in a nucleus-satellite hierarchy (§2.1.4). Each sentence has a core nucleus situation, modified or elaborated on by satellite situations, which can in turn give rise to their own semantic chunks.

Analyzing propositions based on their participants and spatio-temporal location allows us to group or separate fragments of sentences, depending on which situation they refer

to. In particular, the introduction of a Davidsonian eventuality variable provides us with practical means of making the distinction between groups of propositions in the logical representation of a sentence. Throughout the thesis we take semantic chunks to comprise sets of propositions with a common eventuality variable.

Situations and their respective semantic chunks are strongly associated with the presence of verbs, the usual carriers of Davidsonian variables. After considering various forms in which events appear in sentences, we concluded that syntactic clauses provide the best match for the chunking constraints. Eventive noun phrases and attributive adjectives fall short in terms of reliable recognition and granularity.

Granularity appears as a recurring key issue in our discussions of suitable candidates for semantic chunks. The success of chunking relies on the improved processing performance of the target task as applied to smaller fragments of a sentence, compared with the cost of processing it all at once. Both insufficiently small and overly large fragments lead to inefficiencies. For each task and representation there exists a Goldilocks zone of optimal processing efficiency and quality.

Semantic chunking is firmly anchored in the compositionality principle. The connection between semantic chunks and situations emerges as the result of rearranging structural components of propositions, possible because of the premise that the meaning of a sentence comprises smaller meaningful constituents. According to the principle of compositionality, the meaning of a complex expression is determined by its structure and the meanings of its constituents. While individual chunks provide the latter part, the meaning aspect encoded in the structure of the sentence is dictated by functional fragments. They include linking words, such as coordinators and subordinating conjunctions, but also sentence- or clause-level modifiers which scope over multiple chunks (cf. Example 5 in Chapter 1). Any processing of sentences in the chunked format has to not only account for the chunks themselves, but also incorporate the auxiliary information about how the chunks combine to yield the full sentence.

The interaction between chunks can go beyond the relations expressed in functional fragments. Each semantic chunk corresponds to a situation, and within logical representations we associated each with a distinct eventuality variable. Although no eventuality variable should appear in two chunks, no such restriction applies to variables corresponding to other entities. Semantic chunks have to be semantically contained but only to the extent supported by the representation and required by the target task. If constituents of different chunks interact, e.g. through a control relation (cf. §2.1.4) and the interaction is exposed by the representation, we need to be able to recover the relationship after chunking. It can be stored

in the chunk itself or in the associated auxiliary functional fragments, but its existence does not preclude valid chunk creation.

Practical solutions to forming valid semantic chunks are the main focus of this thesis. Although we strive to match the theoretical constraints outlined in this section, it bears remembering that semantic chunking is above all a practically motivated task, adjustable to changing requirements. In the remaining chapters we practically explore its various aspects and investigate to what extent the outlined theory can be satisfied in practice.

2.2 Related work

Semantic chunking has strong foundations in existing research and shares objectives with well-established NLP tasks. The idea of simplifying the processing of complex sentences through a divide-and-conquer approach surfaces repeatedly in many applications. The novelty of semantic chunking lies to a large extent in its flexible objective-oriented definition, which can be considered a generalisation over some of the related tasks discussed below.

2.2.1 Shallow chunking

What we refer to as semantic chunking is distinct from the chunking used as partial parsing. Shallow chunking, as that task is commonly known, identifies consecutive fragments of text which form basic phrases headed by content words, i.e. noun phrases, verb phrases, etc. (Abney, 1992; Jurafsky and Martin, 2009). The result is akin to a flattened syntactic parse tree. Many variants of the task exist. Example 37 is chunked based on the interpretation reflected in the most commonly used dataset: CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000)[6].

(37) [$_{NP}$ He] [$_{VP}$ reckons] [$_{NP}$ the current account deficit] [$_{VP}$ will narrow] [$_{PP}$ to] [$_{NP}$ only £ 1.8 billion] [$_{PP}$ in][$_{NP}$ September].

The fragments created by this approach are even smaller than the pieces we suggested in the beginning of Chapter 1 as a possible outcome of shallow chunking. The CoNLL task lists 10 types of chunks, but it is an unusually broad spectrum. As Tjong Kim Sang and Buchholz (2000) note themselves, the interest of the NLP community focused on noun phrase chunks. These days, shallow chunkers are common tools present in many of the standard NLP software packages, such as OpenNLP[7] or NLTK[8] (Loper and Bird, 2002).

[6] Example 37 comes from Tjong Kim Sang and Buchholz (2000)
[7] https://opennlp.apache.org/
[8] https://www.nltk.org/

The size of shallow chunks is a side effect of assumptions underlying their formation. They are flat and non-recursive, i.e. an NP chunk cannot contain any smaller NPs. None of these restrictions apply to semantic chunks, which are generally larger units, comprising multiple shallow chunks. The granularity of semantic chunks varies from task to task and candidates for chunks can remain merged together if they are too small or not well-formed by the standard of the target task. Depending on the representation used, semantic chunks also have a clear hierarchy with respect to one another, and chunking can therefore be represented as nesting brackets rather than the disjoint brackets used in shallow chunking.

Typically, shallow chunks are defined on the syntactic form of the sentence, while semantic chunks rely on it only to the extent that it is reflected in the chosen representation (§2.1.3). A special case are applications which operate on the surface form of examples, but even then, the syntactic form can be thought of as a filtering criterion for semantic chunks needed by the particular task rather than their defining feature. For example, only semantic chunks with continuous surface representations are retained for the target task of parsing in Chapter 4.

The two types of chunking have distinct goals and are appropriate in different scenarios. Shallow chunking is a technique for partial parsing used when full parsing is not needed or when its cost is too large relative to its gains, while semantic chunking is not a substitute for another operation. Although both tasks can be used to mitigate costs of downstream processing, this objective is the foundation of the definition of semantic chunking. At the same time, the definition allows flexibility in terms of what is the best approach to achieving the goal.

2.2.2 Text simplification

Another set of fragments we proposed in the introduction chapter was a potential output of sentence simplification, which converts a complex sentence or a set of sentences into a set of sentences which are shorter and easier to understand. The aim is to reduce the linguistic complexity of a text, while still retaining the original information and meaning (Siddharthan, 2014)[9]. This is achieved by a combination of pruning a sentence, rearranging it syntactically, and replacing words and phrases with more straightforward equivalents. Sometimes, it can be defined more broadly to include conceptual simplification or summarisation.

The stated aims of simplification and semantic chunking seem similar, but the crucial difference is the target recipient of the output of each task and the extent to which the input can be modified during the processing. In semantic chunking the result is judged by the performance of a downstream target task, while the target judge in simplification is a human receiver of the final text. The simplification process can remove tangential information in

order to highlight the core of the sentence meaning. In contrast, the content of semantic chunks cannot be modified.

Semantic chunking can be used as a component of a more complex simplification system, supporting or replacing a commonly present syntactic simplification module. Since the task of simplification focuses on the meaning of the sentence, self-contained fragments provided by semantic chunks are particularly well-suited as starting points of further transformations.

Split and Rephrase

Narayan et al. (2017) proposed a variant of the simplification task called Split and Rephrase(S&R), allowing only those types of syntactic simplification which preserve the meaning of sentences. The task was further investigated by Aharoni and Goldberg (2018) and Botha et al. (2018), and it isolates two common components of the classic simplification task: splitting and rephrasing. Taking as an input a complex sentence, S&R requires its output to take the form of simpler but still grammatical sentences; e.g. relative clauses have to be converted into main clauses. As a result, the product of semantic chunking can correlate with the splits intended by S&R, but it does not align with the secondary objective of the syntactical well-formedness. Unlike the results of S&R, semantic chunks are subject to flexible well-formedness conditions, dependent on the target task and representation. This adaptability widens the range of their applications. An S&R system with semantic chunking at its core could be one of them, given an additional rephrasing step.

As part of their experiments, Narayan et al. (2017) implemented an S&R system based on the hybrid simplification model by Narayan and Gardent (2014), which combines a probabilistic module for splitting and deletion with a statistical machine translation system for phrase substitution and reordering aspects of the task. The first component is of particular interest to us because it takes as its input a deep semantic representation: Discourse Representation Structure (DRS; Kamp (1981)) as produced by Boxer (Curran et al., 2007), designed to track mental representations of a hearer throughout the discourse. It consists of entities and conditions describing their mentions in the discourse. One type of entities are events associated with thematic roles, such as agent and patient.

Narayan and Gardent's system considers splitting opportunities between event pairs that have at least one core thematic role. The resulting fragments resemble semantic chunks found by the system described in Chapter 3 of this thesis – finite clauses with a syntactic subject-verb form, although due to details of DRS, the simplification splits allow outputs based on non-finite clauses. The splits are chosen from among the candidates based on probabilities learned through the expectation-maximization (EM) algorithm on DRS structures converted to trees. Without the rephrasing module, what the system effectively performs is semantic

chunking on a DRS representation, showing that the task can be viable for deep semantic representations other than *MRS.

A later developed variant of the simplification system (Narayan and Gardent, 2016) makes a step towards an unsupervised approach. Instead of learning the splits based on a dataset of aligned complex and simple sentences, the second version chooses among split opportunities based on a language model of simple sentences alone, taking into account their length and associated semantic patterns. Chapter 5 of this thesis takes the idea of automatic splitting further, as we show how semantic chunks in a complex sentence can be found based on the inherent structure of its deep semantic representation. Like Narayan and Gardent (2014), we also found it useful to treeify the representation (§5.1).

2.2.3 Clause identification

Semantic chunking in its widest scope, which we outlined in §2.1.3, bears resemblance to clause identification.

Huddleston and Pullum (2002, Chapter 2) define a clause as "a syntactic construction consisting (in the central cases) of a subject and a predicate", which excludes many fragments we consider valid semantic chunks. Leffa and Cunha (1998) use a wider definition which includes verb phrases with non-finite verbs, which many classifications treat as phrases. The widest definition of a semantic chunk that we have arrived at in §2.1.3 selects the same type of sentence fragments as this definition of a clause. Notably, the choices made by Leffa and Cunha were directed by practical considerations. The authors posit that clauses ultimately behave as if they were one word. In order to process them we have to be able to identify and classify them according their syntactic function in the sentence, based on what type of word can be used to replace them. We base our generalization across chunking decisions in Section 5.4.2 on a similar assumption.

While Leffa and Cunha (1998) take practical considerations into account when defining clauses, this is not a typical approach, while semantic chunks are defined primarily in terms of their purpose as a means of intermediate processing of long sentences. Not every clause boundary is a chunk boundary suitable for every task, even if the set of clauses and the largest candidate set of semantic chunks overlap significantly.

Clause identification was the CoNLL-2001 shared task (Tjong Kim Sang and Déjean, 2001), but that variant of the task avoids defining the clause in theoretical terms, instead relying on the Penn Treebank bracketing guidelines for annotations (Bies, 1995). Furthermore, while clauses in the CoNLL task and the subsequent systems attempting it rely on hand-defined guidelines and human annotations, we observe that semantic chunks emerge naturally

from the structure of semantic representations such as DMRS (§5.1), reflecting the underlying linguistically designed grammar (§2.3).

2.2.4 Segmentation in text and speech

Semantic chunking aims to isolate semantically contained fragments of a sentence representation in order to process them more efficiently than it is possible for the whole. A similar goal on a higher level is pursued by sentence segmentation, or a sentence boundary disambiguation task, where a longer discourse is divided into smaller meaningful fragments, sentences.

Despite conceptual similarities, practical aspects of the two tasks differ significantly. Sentence segmentation in English largely depends on punctuation marks. Key challenges are the recognition of abbreviations and alternative uses of punctuation, such as emoticons. A commonly used splitting system[10], Punkt tokenizer (Kiss and Strunk, 2006), is an unsupervised model comprising several stages of abbreviation detection and classification. Although the task is no longer a popular field of research and widely considered solved, the increase in the volume of research into both more informal and more technical subdomains of language suggests the need to revisit the task (Read et al., 2012).

A more recent variant of the task, prompted by advances in speech recognition technology, is disambiguation of speech-units in spoken text. The concept of conventional sentences is not well-suited for natural conversation. Instead, speech units are units within the discourse that function to express a complete thought or idea on the part of the speaker, designed to make raw speech-to-text output more useful to downstream processing (Strassel et al., 2005). One can readily draw parallels between semantic chunks and speech units, as they both represent semantically contained fragments useful for processing longer inputs.

Despite potential commonalities between the two types of fragments, speech utterances lie outside of the scope of this thesis as the domain of spoken language leads to unique challenges: presence of disfluencies, additional meaning encoded in prosodic elements of speech, and turn-taking in dialogues, to only mention a few (Caines et al., 2017).

2.2.5 Summarisation

The tasks we discussed so far are related to semantic chunking through similarity. On the other hand, summarisation is an example of a semantically focused task which can benefit from semantic chunking. Its goal is to produce an abridged version of a document or a set of documents which contains the important or relevant information. The task is performed

[10]For example, as part of the popular NLTK library, www.nltk.org (Loper and Bird, 2002).

routinely by humans, but automated systems struggle to reach comparable results. Two approaches to summarisation are a) extractive, where the new text is composed of selected fragments of the original, and b) abstractive, where an entirely new text is generated from the information contained in the input text. The latter approach offers better information packaging, but requires deeper levels of NLP text analysis and knowledge representation (Fang et al., 2016).

An example of an abstractive summariser relying on the concept of a proposition is the system by Fang and Teufel (2014, 2016) rooted in a model of human comprehension by Kintsch and Van Dijk (1978). An input text is processed as a collection of propositions, which are copied into memory or "forgotten", based on relationships between them, with the surviving propositions forming the output summary. The core of the summariser uses the Stanford parser (Klein and Manning, 2003) so that propositions reflect dependency relations. For example, Example 38 gives rise to the propositions in 38ab (Fang et al., 2016).

(38) The discovery of the element revolutionised fire-lighting.

a. *revolutionised(the discovery, fire-lighting)*

b. *of(the discovery, the element)*

The resulting notion of a proposition is more fine-grained than the meaning units we adopt for semantic chunking, and it is based on purely syntactic relationships between words as indicated by the choice of proposition heads, e.g. *of* in 38b. Fang et al. (2016) enhance the summariser with a natural language generation step, linking propositions to DMRS subgraphs of original sentences. Propositions in that form can be converted to natural sentences by one of the *MRS processors (§2.4). The resulting summaries are made up of simplified original sentences, comprising only the parts corresponding to the selected propositions. Semantic chunking can facilitate finding appropriate subgraphs for each proposition.

The concept of a proposition is also useful in the context of summarisation for comparing informational content of different summaries of the same text. An example of a unit of informational content is a factoid (Teufel and van Halteren, 2004; van Halteren and Teufel, 2003). The meaning of a sentence is a union of factoids. For example, Example 39 (from van Halteren and Teufel (2003)) consists of the factoids in 40.

(39) The police have arrested a white Dutch man.

(40) a. A suspect was arrested

b. The police did the arresting

c. The suspect is white

d. The suspect is Dutch

e. The suspect is male

In a similar vein, Nenkova et al. (2007) proposes to organise summaries into a pyramid structure made up of factoid-like summary content units (SCUs).

Like semantic chunks, factoids are compositional, but their granularity is determined by the dataset rather than the task or representation properties. If a set of simple factoids always occurs together in the data, they are combined into a single, more complex factoid. The resulting statements often do not have a one-to-one correspondence to sentence fragments, e.g. the word *suspect* used in the example factoids above never appears in the original sentence. Factoids are intended as a tool for human annotators, rather than to facilitate automated processing.

The propositions forming factoids can be created from all parts of speech, including descriptive adjectives and nouns. For example, two arguments of the verb *arrested* yield two separate factoids 40a and 40b. As discussed in §2.1.2, such an interpretation of meaning units would result in excessive proliferation of semantic chunks, with individual chunks often corresponding to single words. Incorrect choice of the granularity level obstructs the primary goal of reducing the complexity of input examples. As a result, we recognize only one semantic chunk in the entire example 39.

2.2.6 Machine translation

Machine translation is one of the tasks most affected by the length of the sentence and it was a major motivation for the creation of semantic chunking as an intermediate task. Producing a translation requires dealing with all aspects of complexity of multiple languages and differences between them. It is common for statistical and neural machine translation (SMT and NMT) systems to exclude sentences longer than 30 words from their training sets for reasons of computational efficiency, e.g. Cho et al. (2014).

Hierarchical phrase-based translation, introduced as Hiero by Chiang (2005, 2007), extends phrase-based translation to allow hierarchical phrases containing subphrases. The model is a synchronous context-free grammar of rewrite rules between source and target sentences. Subphrases are expressed by non-terminal symbols, and the full translation is built up by substituting partial translations of subphrases into the phrases with non-terminals. Our approach to semantic chunking described in Chapter 5 was inspired by Hierarchical Semantic Statistical Translation (HSST) and Realization (HSSR) system, developed by Horvat et al. (2015) and Horvat (2017). We discuss it in more detail in §5.3.1.

A common technique in machine translation is clause-level reordering, designed to tackle word order differences between languages. It has a long history of employing the divide-

and-conquer principles we propose for leveraging semantic chunks in the processing of long sentences. For example, in a translation system by Watanabe et al. (2003) a source sentence is first shallowly chunked (§2.2.1), then each chunk is translated into a target language with local word alignments, and finally the translated chunks are reordered to match the target language constraints. Ishiwatari et al. (2017) designed a chunk-based decoder for a neural machine translation system, consisting of a chunk-level and a word-level decoder. The former generates a chunk representation containing global information, which the latter uses as a basis to predict the words inside the chunk.

The systems mentioned above and other similar ones rely on the small size of shallow chunks, which are the length of simple phrases. As a consequence, representations of shallow chunks are more akin to multi-word expression representations (e.g. Legrand and Collobert (2016)), due to their size and scope, while distributed representations of semantic chunks would face challenges similar to those associated with sentence-level distributed representations (e.g. Hill et al. (2016)).

Another variant of the divide-and-conquer approach focuses on specific types of fragments. Koehn and Knight (2003) created a dedicated noun phrase translation subsystem. Sudoh et al. (2010) focus on embedded clauses, such as relative clauses, in Japanese-English translation. The embedded clauses are translated separately and used to reconstruct final translations with the help of surface non-terminals. Example 41 from Sudoh et al. (2010) illustrates their approach:

(41) a. John lost the book that was borrowed last week from Mary.

b. john wa senshu mary kara kari ta hon o nakushi ta.

c. i. john wa _s0 hon o nakushi ta . → John lost the book _s0 .

ii. senshu mary kara kari ta → that was borrowed last week from Mary

The relative clause in 41c-ii is translated separately and the result is substituted into the main clause in place of the non-terminal *s0*. We employ a similar placeholder technique in our realization experiments (§3.6).

Pouget-Abadie et al. (2014) apply the divide-and-conquer principle specifically to address the lower performance of NMT systems on long sentences. They take a step towards semantic chunking by automatically segmenting an input sentence into phrases that can be easily translated by the neural network translation model. The sentence fragments are selected to maximise the sum of confidence scores of the fragment translations. We can recognise one of the key principles of semantic chunking in the tailoring of the output to the needs of the target task. The authors observe overall improvement in the translation quality, but only when the segments are well-formed sentential clauses. Splitting the sentence in a way which separates

complements from their predicates is one of the highlighted causes for mistranslation. This is in line with our intuition that the fragments used for the divide-and-conquer approach need to be coherent and contained semantically.

As the survey of machine translation approaches presented above illustrates, challenges posed by machine translation resonate strongly with the design of semantic chunking as a task. It offers a principled and flexible way of reducing the complexity of sentences without affecting the semantic transfer between languages. At the same time it provides a common framework for the various techniques already used in different fields and approaches.

2.3 *MRS and the ERG

In §§2.1 and 2.2 we introduced theoretical foundations of semantic chunking and compared it to existing tasks. Before moving on to the implementation, we introduce the framework of our experiments.

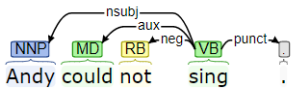
This thesis focuses on exploring semantic chunks with Dependency Minimal Recursion Semantics (DMRS). A DMRS of a sentence is a graph of its semantic dependency structure (Copestake, 2009). It originated as a more readable form of Minimal Recursion Semantics (MRS; Copestake et al. (2005)) — a flat representation for compositional semantics. MRS and its related representations are referred to collectively as *MRS. We focus on DMRS, but since for now there is no way to produce DMRS parses directly without conversion from MRS, a certain level of understanding of the original MRS format is required. The conversion from MRS to DMRS is a fully deterministic, one-to-one mapping.

*MRS is the base formalism of many grammars, including the English Resource Grammar (ERG) (Flickinger, 2000, 2012; Flickinger et al., 2014), a broad-coverage, symbolic grammar of English, developed as part of the DELPH-IN initiative[11] and LinGO[12] project (Copestake and Flickinger, 2000). Its main goal, other than broad coverage, is efficient processing. The work done in this thesis furthers this aim in particular. The *MRS produced by the ERG are sometimes referred to as ERS. New versions of the grammar are being continuously developed. Throughout this thesis, we work with the ERG 1214 version.

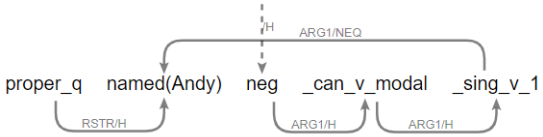
The deep semantic representation is produced automatically by a parser on the basis of a linguistically motivated grammar, which in this framework can be considered a form of one-off intensive annotation effort (Bender et al., 2015). Thanks to this set-up, we are able to analyse new sentences on the fly without the additional involvement of human annotators. The framework's deeply embedded compositionality guarantees our ability to readily identify

[11] http://www.delph-in.net/

[12] Linguistic Grammars Online, website: lingo.stanford.edu



(a) Stanford dependencies for Example 42. Parsed with http://corenlp.run.



(b) DMRS for Example 42.

Fig. 2.2 Syntactic and semantic dependencies of Example 42.

constituents, as the linguistic notions encoded in the grammar are reflected in the machine-readable structure of *MRS. As a result, fragments suitable to act as semantic chunks can be reliably explored and detected automatically (Chapters 3 and 5), supporting the flexible nature of the task design. At the same time, the *MRS analyses are strongly guided by syntax. The connection between the meaning representation and the surface form of sentences allows us to experiment with the transfer of semantic chunks between the two representations. This opens up a highly desirable opportunity for semantic chunking of sentences based solely on their strings (Chapter 4).

We touched upon differences between syntactic and semantic dependencies in the introduction to Chapter 1. One important factor we discussed is the representation of scopal relationships between predicates, e.g. how various adverbs interact with other sentence elements (cf. Fig. 1.2 and 1.3). Although two elements interdependent in a syntactic parse tend to also interact semantically, especially in the compositional representation such as *MRS, the direction of dependency is often swapped. For example, let us consider the syntactic and semantic dependencies for Example 42, shown in Figure 2.2.

(42) Andy could not sing.

Every single syntactic arc in Figure 2.2a starts at the verb *sing*, which has subject, auxiliary and negation dependencies. In the meantime, the same verb in the DMRS (Fig. 2.2b) has only one semantic dependency, labelled ARG1/NEQ, corresponding to a thematic agent role. Both the negation and modal verb modify the main verb rather than being its dependents, which reflects the scopal relationships between the predicates. The distinction allows us to differentiate between two interpretations of the sentence:

(43) Andy was really bad at singing.

(44) Andy was capable of not singing.

The DMRS in Figure 2.2b represents the first scopal resolution of Example 42. The other variant would see the order of links between the negation and modal verb reversed (cf. §2.3.2).

Information	Value
predicate	`_sensible_a_for`
span	39:47
label (LBL)	h25
intrinsic variable (ARGO)	e26
node properties	SF: prop, TENSE: untensed, MOOD: indicative, PROG: -, PERF: -
arguments	ARG1: x21

Table 2.1 The full information about the predicate `_sensible_a_for` present in the example MRS in Fig. 2.3.

2.3.1 Elementary predications and nodes

The basic unit of *MRS is an **elementary predication** (EP), which corresponds to a relation and its associated arguments. Each EP has a label called **handle**, a **predicate**, and a list of arguments.[13]. One of the arguments, ARG0, is a unique **characteristic variable** (Copestake, 2009) or an **intrinsic argument**[14].

An MRS example is shown in Figure 2.3. Each EP is displayed in an individual line of the RELS section, although we omit some properties from Figure 2.3 for clarity. The example of full information about any given predicate included in the analysis is shown in Table 2.1 for the predicate `_sensible_a_for`. The graph in Figure 2.4 is the DMRS equivalent of the MRS example. In DMRS all the lexical and grammatical information about a given EP is contained in a node with a unique identifier (nodeid). Below we will use the terms node and EP interchangeably.

Predicates in the ERG come in two flavours: **real** and **grammar**. Real predicates correspond to particular lexemes and they have a lemma, a part-of-speech tag and a sense associated with them: `_lemma_pos_sense`, e.g. `_example_n_of`. Grammar predicates are a fixed set of over one hundred predicates (Horvat, 2017) with functionality varying from representing proper nouns to introducing subordinated clauses. They consist of a predicate name and a `carg` value if they have a surface representation. For example, `neg` introduces negation, while `named('Andy')` represents the proper name *Andy* stored as the predicate's `carg` attribute.

Each real predicate included in the grammar has one or more entries in the grammar's lexicon file[15]. The entries specify the base orthography of a predicate and its lexical type related to its subcategorization. Although each entry has a unique grammar-internal name,

[13]http://moin.delph-in.net/DelphinTutorial/Formalisms
[14]http://moin.delph-in.net/ErgSemantics/Basics
[15]The lexicon file for the ERG 1214: http://svn.delph-in.net/erg/tags/1214/lexicon.tdl.

```
[ LTOP: h0 INDEX: e2
RELS:
< [ proper_q LBL: h4 ARG0: x3 RSTR: h5 BODY: h6 ]
  [ named LBL: h7 CARG: "Andy" ARG0: x3 ]
  [ neg LBL: h1 ARG0: e9 ARG1: h10 ]
  [ _can_v_modal LBL: h11 ARG0: e2 ARG1: h12 ]
  [ _possible_a_for LBL: h13 ARG0: e14 ARG1: h15 ]
  [ _come_v_up LBL: h16 ARG0: e17 ARG1: x3 ARG2: u18 ]
  [ _with_p LBL: h16 ARG0: e19 ARG1: u20 ARG2: x21 ]
  [ _a_q LBL: h22 ARG0: x21 RSTR: h23 BODY: h24 ]
  [ _sensible_a_for LBL: h25 ARG0: e26 ARG1: x21 ]
  [ _example_n_of LBL: h25 ARG0: x21] >
HCONS: < h0 qeq h1 h5 qeq h7 h12 qeq h13 h10 qeq h11
         h15 qeq h16 h23 qeq h25 > ]
```

Fig. 2.3 MRS representation of *Andy couldn't possibly come up with a sensible example.*

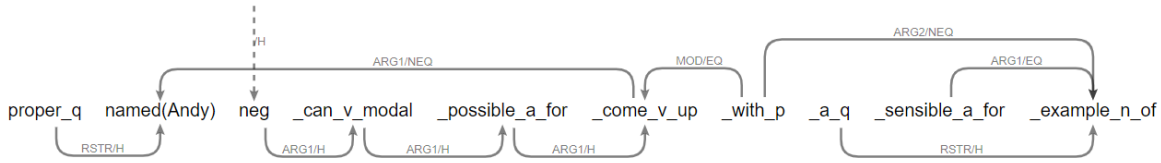


Fig. 2.4 DMRS representation of *Andy couldn't possibly come up with a sensible example.*

the predicates can appear in multiple entries with different lexical types. For example, the predicate `_examine_v_1` has two entries: `examine_v1` and `examine_v2`. The former has the lexical type `v_np_le` and applies when the predicate takes an object (Ex. 45), while the type of the latter entry links it to scenarios with a clausal interrogative complement (Ex. 46):

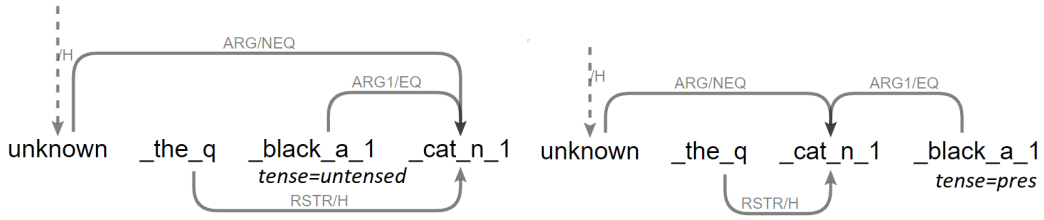
(45) The cat examined the fish.

(46) The cat examined whether the fish was fresh.

The *MRS analyses of the two examples represent *examined* with the same predicate, and it is up to the parser to distinguish between them at runtime.

When a lexical item cannot be identified with any of the grammar predicates, the ERG assigns it a custom predicate, with a lemma made up of the unknown token and its Penn Treebank PoS tag, e.g. `_bamboozled/VBN_u_unknown`.

Other properties of a node are determined by the type of its characteristic variable: **instance** (marked as `x`) or **eventuality**/event (`e`). We will refer to the nodes associated with each type as **instance nodes** and **event** or **eventuality nodes** respectively. Quantifiers do not have a variable of their own. In general, instance nodes are noun-like with gender, number



(a) DMRS of *the black cat* (b) DMRS of *the cat which is black*.

Fig. 2.5 The tense of copular constructions with adjectives and prepositional phrases is marked on the meaning-bearing predicate.

Property	Meaning	Possible values
tense	Verb tense	untensed; *tensed*: present, future, past
perf	Perfective aspect	boolean
prog	Progressive aspect	boolean
mood	Grammatical mode	indicative, subjunctive
sf	Sentence force	*prop-or-ques*: proposition, question; command (imperative)

Table 2.2 Properties of event nodes in ERG1214. Values in italics are underspecified, i.e. they can be used when the distinction between the finer values is not needed.

and person, while eventuality nodes have verb-like properties, listed in Table 2.2 for ERG 1214. They are associated with the concept of eventuality introduced in §2.1.2, where we discussed the relationship between semantic chunks and situations.

In §2.1.3 we discussed suitability of clauses as syntactic representations of chunks. In particular, we took the presence of a verb as a good indicator of a new situation being introduced. The ERG follows an analysis in which eventualities correspond to not only verbs, but other parts of speech as well, both in predicative and attributive constructions (§2.1.2). Copular verbs do not have their own nodes when paired with adjectives or prepositional phrases. Instead, the node of the main predicate of the construction, e.g. `black` in Figure 2.5b, has the tense marked in its properties.

The two DMRS in Figure 2.5 represent an attributive adjective and a relative clause. They are identical apart from the value of the tense feature on the `black` node. The order of nodes in the visualisation reflects the surface order but is not inherent to the DMRS itself, while the `unknown` predicate marks a fragment analysis (§2.3.5).

EPs created from a parsed sentence have character spans associated with them (cf. Table 2.1). Described by two values, `cfrom` and `cto`, the character spans link each EP to a surface string. When an EP is based on a lexeme, `cfrom` and `cto` point to the starting and ending indices of the lexeme string within the sentence string. Unlike real predicates, some grammar predicates are introduced syncategorematically, i.e. through grammar rules rather

than by lexical items. As a result, their character spans are more arbitrary and depend on the details of the grammar.

We consider two nodes to be equal if they have the same predicate, `carg` attribute, variable and its associated properties. A node can be also less or more specific than another node. For example, a node with the `tensed` value for the tense property is less specific than a node with the `past` value.

2.3.2 Arguments and links

*MRS is a compositional representation (§2.1.4), in which the composition is achieved through a system of hooks and slots (Copestake et al., 2001). Each EP has a unique characteristic variable ARG0 and potentially a list of other arguments numbered ARG1, ARG2, etc. The names of arguments are unique, but not thematic. Instead, they reflect the syntactic position of arguments as formalized in HPSG[16].

Arguments express hook-slot relationships between predicates. Their values can be either variables of other EPs in the MRS (e.g. `x3`) or their handles (e.g. `h11`). The hook of one EP, its variable or handle, fills the argument slot of another EP. The handle arguments are **scopal**, as opposed to **non-scopal** variable arguments. We refer to nodes with at least one scopal argument as **scopal operators**[17].

Let us consider the EPs with predicates `_come_v_up` and `_possible_a_for` from the example MRS in Figure 2.3:

```
[ _possible_a_for LBL: h13 ARG0: e14 ARG1: h15 ]
[ _come_v_up LBL: h16 ARG0: e17 ARG1: x3 ARG2: u18 ]
```

The `_come_v_up` EP has one non-scopal argument `x3`, which is the characteristic variable of the `named` EP[18]. On the other hand, the `_possible_a_for` EP has a scopal argument `h15`, which does not match the label of any EP in the example. Scopal arguments are resolved by the HCONS list of *qeq* relations at the bottom of the MRS. In the case of `possible` we can see that `h15` *qeq* `h16`, and `h16` is the handle of the `_come_v_up` EP. This indicates that the adverb scopes over the verb.

The representation of scopal relations as handles in *MRS has its origins in conventional predicate calculus and facilitates one of the main properties introduced by the formalism: underspecifiability with respect to the quantifier scope, which means that semantic distinctions

[16] http://moin.delph-in.net/ErgSemantics/Essence
[17] http://moin.delph-in.net/ErgSemantics/Basics
[18] The ARG2 is an uninstantiated variable in this use of the verb

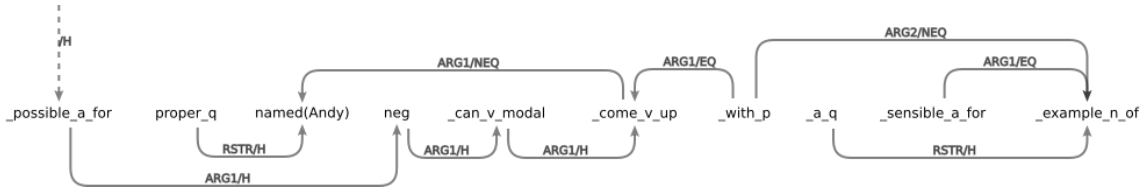


Fig. 2.6 DMRS of *Possibly, Andy couldn't come up with a sensible example.*

```
[ LTOP: h0 INDEX: e2
RELS:
< [ proper_q LBL: h4 ARG0: x3 RSTR: h5 BODY: h6 ]
 [ named LBL: h7 CARG: "Andy" ARG0: x3 ]
 [ neg LBL: h1 ARG0: e9 ARG1: h10 ]
 [ _can_v_modal LBL: h11 ARG0: e2 ARG1: h12 ]
 [ _possible_a_for LBL: h13 ARG0: e14 ARG1: h15 ]
 [ _come_v_up LBL: h16 ARG0: e17 ARG1: x3 ARG2: u18 ]
 [ _with_p LBL: h16 ARG0: e19 ARG1: u20 ARG2: x21 ]
 [ _a_q LBL: h22 ARG0: x21 RSTR: h23 BODY: h24 ]
 [ _sensible_a_for LBL: h25 ARG0: e26 ARG1: x21 ]
 [ _example_n_of LBL: h25 ARG0: x21] >
HCONS: < h0 qeq h13 h5 qeq h7 h12 qeq h16 h10 qeq h11
         h15 qeq h1 h21 qeq h23 > ]
```

Fig. 2.7 MRS representation of *Possibly, Andy couldn't come up with a sensible example.*

associated with the scope can be left unresolved (Copestake et al., 2005). Let us alter the running example slightly:

(47) (original) Andy couldn't possibly come up with a sensible example.

(48) Possibly, Andy couldn't come up with a sensible example.

The meaning of the two sentences differs only because of the scope of the adverb *possibly*. A simplified conventional logic might yield the following representations:

(49) `neg(can(possible(come_up_with(Andy, x) & example(x) & sensible(x))))`

(50) `possible(neg(can(come_up_with(Andy, x) & example(x) & sensible(x))))`

When we compare the original MRS in Figure 2.3 with the new variant in Figure 2.7, we notice that they are identical apart from the HCONS list. DMRS graphs combine the RELS and HCONS sections of the MRS into links, so that the scope resolution for each reading can be seen more clearly in their DMRSs in Figures 2.4 and 2.6. The links in the two versions chain in a way that matches conventional bracketing:

$$\texttt{neg} \xrightarrow{ARG1/H} \texttt{can} \xrightarrow{ARG1/H} \texttt{possible} \xrightarrow{ARG1/H} \texttt{come up}$$
$$\texttt{possible} \xrightarrow{ARG1/H} \texttt{neg} \xrightarrow{ARG1/H} \texttt{can} \xrightarrow{ARG1/H} \texttt{come up}$$

This is an example of the improved readability of DMRS representation compared with its MRS predecessor.

The comparison with a bracketed notation illustrates that scopal operators form a hierarchy. This is the result of the assumption that a well-formed MRS can be resolved to a conventional logical representation through the handle constraints (Copestake et al., 2005). Two scopal operators cannot both take the same handle as its argument. If an operator P is applied to a handle which is already modified by a scopal operator Q, P has to either outscope Q or become the argument of Q, taking over its original scope. We discuss the scopal hierarchy in more detail in Chapter 5.

A special class of scopal operators are quantifiers, associated with a unique argument structure. In DMRS they can be recognised by their unique RSTR/H link. Development of *MRS was guided by the need to allow the scope to be ignored when irrelevant, but retrievable when required (Copestake et al., 2005). Thanks to that design, quantifiers are excluded from the general scopal hierarchy and can be safely ignored for the purpose of semantic chunking. Our use of the term scopal operator in the remainder of the thesis reflects this exclusion.

Links in DMRS combine arguments and handle constraints. Each node is connected to its arguments via links, and different types of arguments have different link labels. Each link has a start node, an end node and a **label** in the form A/B, consisting of an argument name (A), and one of the following four link types (B):

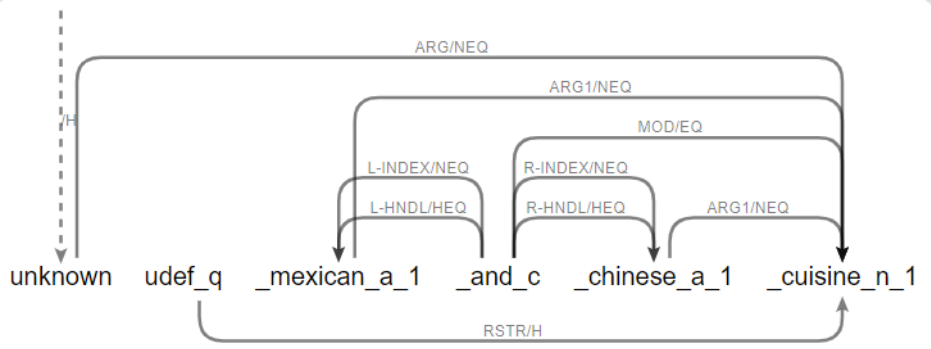
EQ Connected nodes share a handle.

H Indicates a scopal argument.

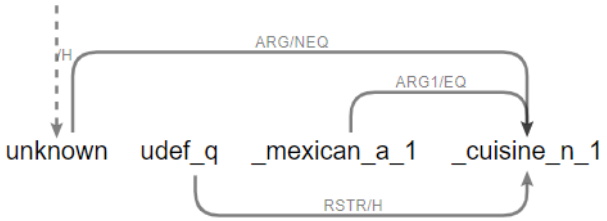
HEQ A hybrid of the two. The value of the argument is scopal in the sense that it is a handle, but it is directly a handle of another EP, not connected to it through a *qeq* link. This intuitively represents a tighter connection between a node and its argument than a normal H link.

NEQ None of the above.

Apart from directed argument links, a DMRS graph can also have undirected EQ links, represented as MOD/EQ or -/EQ. Handle equality is transitive, so that if a node A links to a node B with ARG1/EQ and a node C links to the node B with ARG1/EQ, the nodes A and B share the handle as well. MOD/EQ links are introduced whenever the argument



(a) DMRS of *Mexican and Chinese cuisines*



(b) DMRS of *Mexican cuisine*

Fig. 2.8 Coordination of attributive adjectives requires an undirected MOD/EQ link.

links and transitivity do not account for all shared handles and scope seniority. One type of construction where undirected links make an appearance is coordination of attributive adjectives (Fig. 2.8a compared with Fig. 2.8b without coordination).

2.3.3 Top, index and central nodes

There are two elements of MRS which we have left undiscussed so far: LTOP and INDEX (cf. Fig. 2.3). In practice, LTOP answers the question: if the phrase represented by the MRS were to become a scopal argument of another EP, which handle would be its hook? The scopal hierarchy of operators ensures that only one label is the answer, and it belongs to the topmost operator in the hierarchy. We refer to the node corresponding to the LTOP value as the **top node** of the DMRS. The value of INDEX is the characteristic variable of the EP which determines grammatical properties of the phrase, and corresponds to the **top index node**. In the absence of scopal operators the same node plays the role of the top and top index node. When it becomes an argument of a scopal operator, the two roles diverge: the role of the top node is taken over by the operator. The behaviour is illustrated by the right coordinate of the example in Figure 2.9.

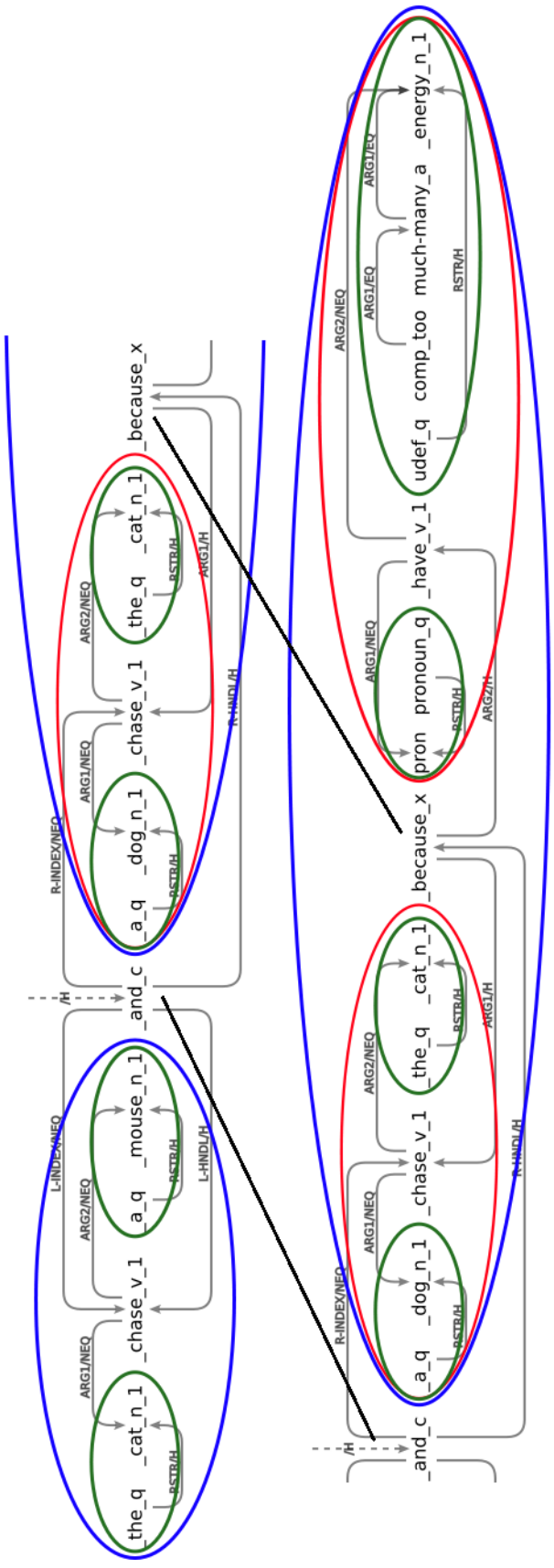


Fig. 2.9 DMRS for *The cat chased a mouse and a dog chased the cat because it had too much energy.* The large graph was split to fit on the page, and the black lines indicate the alignment between the two parts. The coloured ellipses mark semantic subgraphs.

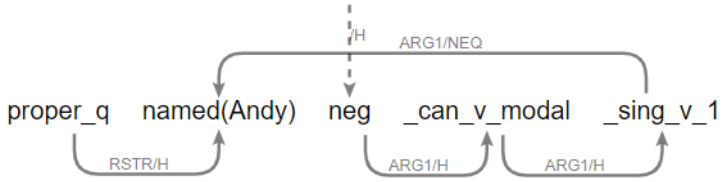


Fig. 2.10 DMRS of *Andy can't sing.*

For the purpose of this thesis we would like to introduce a further distinction. Let us consider the DMRS for Example 51 (Fig. 2.10):

(51) Andy can't sing.

The top node is the negation `neg`, while the grammatical properties of the sentence, such as tense, are determined by the top index node `_can_v_modal`. Intuitively, however, the semantically most important node is the verb `_sing_v_1`. We will refer to such key nodes as **central nodes**. A central node is the lowest node in the graph's scopal hierarchy when the operators are unary[19].

Coordination is a special relation in the ERG1214 in two ways: a) it replaces the central node as a top index, b) it reserves separate argument slots for the index and handle of its coordinates: L-HDNL, R-HNDL, L-INDEX and R-INDEX. In Figure 2.9 the two coordinates correspond to clauses: *the cat chase a mouse* (L) and *a dog chased the cat because it had too much energy* (R). The left coordinate is straightforward. The central node with the verb predicate `_chase_v_1` is both the top and top index node of the subgraph. In contrast, the right coordinate is a complex clause for which the top node is a scopal operator. The subordinating conjunction `_because_x` takes two scopal arguments, one for the top of the main clause (*a dog* ***chased*** *the cat*) and the other for the top of the subordinate clause (*it* ***had*** *too much energy*). The disparity between this coordinate's top and top index nodes is highlighted by the coordination links: R-HNDL, leading to the scopal operator `_because_x`, and R-INDEX, pointing at the main verb of the main clause `_chase_v_1`.

2.3.4 Semantic subgraphs

The compositionality of DMRS means that the representation of a full sentence is composed of subgraphs representing its constituents, as annotated in Figure 2.9. The term subgraph generally denotes any graph formed from a subset of vertices and edges of another graph. When referring to DMRS, it is useful to narrow the definition down and distinguish **semantic**

[19] We discuss other cases in §5.1.

subgraphs[20] as subgraphs of a DMRS which themselves are well-formed fragments or constituents of the sentence, such as phrases and clauses. Semantic subgraphs of the example from Figure 2.9 are marked by coloured ellipses.

A semantic subgraph typically consists of a central node (§2.3.3) with its arguments and modifiers. Each subgraph has its own top node, which is the target of links from scopal operators higher in the hierarchy. In fact, we can think of scopal operators as acting on entire subgraphs, rather than individual nodes. Subgraphs centred around nodes with event variables are sometimes referred to as situations[21], which backs up our assumptions regarding the form of semantic chunks in DMRS (§§3.1, 5.1).

Shared handles

Each EP has a unique intrinsic variable, but a handle can be shared among multiple EPs. The most common scenario where it happens is intersective modification, e.g. *a sensible example* in the MRS in Figure 2.7 (p. 37). Both `_sensible_a_for` and `_example_n_of` have the label *h25*. When we convert an MRS to a DMRS, EPs become nodes and scopal relationships are expressed by links. The number of links outgoing from an operator matches its number of arguments, which means we need to choose which node among those with the correct handle becomes the sole target of the scopal link. Finding the correct target is equivalent to finding the top node of the argument subgraph, but when handles are shared, the top node is not always obvious. Copestake (2009) gives two principles for head selection. If one of the candidate nodes is already an argument of the operator, choose that node. Otherwise, choose the node which does not share a label with any of its arguments. The `pydelphin` library (Goodman (2019), §2.4) further prioritizes grammatical predicates as scopal link targets. The same principles apply for identifying the LTOP node of the whole DMRS.

These rules are sufficient to deal with most scenarios, but often break down in the presence of undirected EQ links (§2.3.2). The inconsistencies disrupt the scopal hierarchy of DMRS graphs. For example, Figure 2.11 shows the DMRS of the sentence:

(52) In International Scientific Vocabulary semantics is also called semasiology.

The heuristic rule which gives precedence to grammar predicates assigns the role of the top to the `focus_d` node. The adverb `_also_a_1` would be a more suitable choice because, in its absence, the top index `_call_v_name` becomes the new top, while, if we remove the leading prepositional phrase, *also* retains its status as the top.

[20] We borrow the term from Horvat (2017), although slightly altering its usage (cf. §5.3.1).
[21] The DELPH-IN Wiki page http://moin.delph-in.net/ErgSemantics/Basics

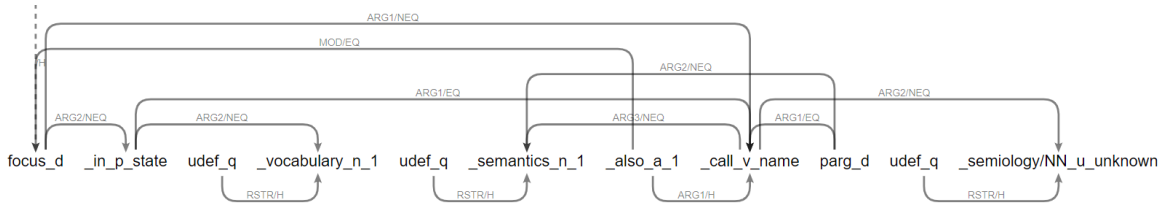


Fig. 2.11 A simplified DMRS of *In* [*International Scientific*] *Vocabulary semantics is also called semasiology.* The nodes corresponding to the words in brackets were removed for better readability.

Adding a few extra heuristics addresses many such discrepancies. For example, we verify that all non-quantifier scopal links lead to nodes with an event variable.

2.3.5 Fragments

In the ERG fragments are constituents which are well-formed utterances when considered in context but not full sentences. They are interpreted as arguments or modifiers of an implied `unknown` predicate[22]. We have already encountered them in Figures 2.5 and 2.8. In some diagrams we ignore the extra node for simplicity. The `unknown` predicate is the top node and the top index, but since it is semantically empty, we usually use the terms to refer to nodes in the remainder of the subgraph instead.

2.3.6 Situations in Head-Driven Phrase Structure Grammar (HPSG)

The English Resource Grammar is developed within the framework of Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag (1987, 1994)), which fully embraces situation semantics (§2.1.2). HPSG is an example of a unification-based theory, in which linguistic objects are analyzed in terms of partial information structures that can be recursively embedded (Pollard and Sag, 1987). Lexical entries, grammar rules, context and general well-formedness principles provide further constraints on the structures.

HPSG integrates syntactic and semantic aspects of grammatical theory under the assumption that neither can be well understood in isolation from the other (Pollard and Sag, 1987). In the original formulation, semantic contents are analyzed as individuals, relations, roles, situations and circumstances. Individuals play roles in relations, which are syntactic but also assign semantic roles. The meaning of a sentence is then the result of unifying pieces of information from constituents. A general principle in HPSG's semantics requires that the semantic content of a phrase be unified with that of its head daughter, i.e. a constituent which

[22]http://moin.delph-in.net/ErgSemantics/Fragments

determines many of the syntactic properties of the phrase as a whole. In DMRS it is typically the index of a subgraph. All operations in HPSG are performed on feature structures, which represent constraints by specifying values for various attributes. The whole framework is recursive and values of attributes can themselves be feature structures.

The semantic content of declarative sentences in the variant of HPSG proposed in Pollard and Sag (1987) is a feature structure of type *circumstance*. A situation, used in the sense of Barwise and Perry (1983) (§2.1.2), supports specific circumstances when the circumstances are facts of the situation. Ginzburg and Sag (2000) expand the ontology to account for all major types of finite clauses in English. They classify phrases not only in terms of the presence and type of head daughter, but also in terms of 'clausality'. Clauses appear in phrases which are 'communicatively complete' – a notion relevant to our concept of contained semantic chunks.

Following Ginzburg and Sag (2000), early versions of the ERG during the LOGON project[23] (Lønning et al., 2004) included *messages* as a type of grammar predicates, e.g. `prpstn_m` (Copestake et al., 2005; Flickinger, 2000). A message can be a proposition, question, outcome, or a fact, and it is the context of the communicatively complete phrases. Messages consist of *states of affairs* (*soas*), which, in turn, are semantic contents of verbs. In order for a soa to be an independent utterance, it has to be embedded within a clausal construction. According to Ginzburg and Sag's grammar, that is possible only for finite and *to*-infinitival clauses, which is a narrower interpretation than the one we assumed in §2.1.3.

The main problem encountered during the development of messages in the ERG was a lack of clear cut-off for what constitutes a message, which led to their proliferation and subsequent abandoning of the approach. We have touched upon the problem of granularity in §2.1 with the discussion of the Goldilocks zone in chunking. Semantic chunking bypasses the need for a theoretical threshold by allowing practical constraints on what constitutes valid chunks. The level of granularity of semantic chunks has to provide a computational advantage to target tasks and can vary between tasks and representations.

2.4 *MRS tools

The ERG (Flickinger (2000, 2012); Flickinger et al. (2014)) is a bidirectional grammar, developed as part of the LinGO project and DELPH-IN consortium, within the HPSG framework. Its primary semantic representation is *MRS, which we discussed in detail in §2.3. There exist several processors, which parse sentences into MRSs and generate surface forms from MRS representations using chart generation. Although in our work we focus

[23] Website: www.emmtee.net

on English, the ERG is only one of the existing DELPH-IN grammars spanning multiple languages. They all use *MRS as their representation, and the concepts we discuss in this thesis can be adapted to suit their needs.

The experiments described in this thesis rely on ACE[24]. We used the pre-compiled grammar file for the ERG 1214, unless specified otherwise. Other parsers and generators operating within the DELPH-IN framework are LKB (Copestake, 2002) and PET[25]. The former is a full grammar and lexicon development environment, trading additional functionality for processing speed when compared to the alternatives. More recent developments in *MRS processors explore the potential of neural network parsing (Buys and Blunsom, 2017) and realization (Hajdik et al., 2019).

2.4.1 Realization

Throughout this thesis we refer to realization or generation as the task which aims to produce a surface form of a sentence from its semantic representation. Often the task definition includes the creation of the semantic content, for example in dialogue systems. Our experiments start with ready MRS structures, typically obtained by parsing, but the same principles apply for analyses constructed in different ways. In fact, §§3.6 and 5.4 include examples of generation from artificially constructed *MRS structures.

The ERG processors such as ACE use the chart generation algorithm exploiting the bidirectional nature of the grammar. The first step is looking up predicates in the grammar lexicon (§2.3.1), together with any potential edges in which they can participate, based on the grammar rules. The generator then attempts to create a syntactic structure whose corresponding MRS matches the input MRS. If the input cannot be linked to existing grammar rules, the generation fails. The complexity of the input analysis can in practice increase the computational cost of such generation exponentially (Carroll et al., 1999; White, 2004). At the same time, the process is sensitive to the well-formedness of its inputs because of the close relationship between chart generation and parsing. This makes realization a good target task to test the quality of semantic chunks (cf. §3.6).

The reliance of realization on the grammar means that out-of-grammar predicates are an issue. The ACE generator does not currently have a mechanism to cope with unknown predicates, but they can be parsed and assigned a part-of-speech tag (§2.3.1). In realization experiments (§3.6, 5.4) we rely on a bypassing technique developed jointly with Horvat (2017). Before an example is input to the generator, each unknown predicate is replaced with a known predicate with the same part-of-speech tag. Afterwards, the known surface form

[24]Woodley Packard's Answer Constraint Engine, version 0.9.25, http://sweaglesw.org/linguistics/ace/
[25]http://pet.opendfki.de/

of the substitute predicate is replaced in the realization string with the surface form of the original unknown predicate. For example, any unknown singular noun predicate is converted into a substitute predicate `_chocolate_n_1`, which realizes as *chocolate* and is grammatical in both countable and uncountable scenarios.

Furthermore, the generator does not achieve full realization coverage, even though it uses the same grammar as the parser which produced the representations. Some lexical items, such as infinitival *to*, are semantically empty according to the ERG analysis, i.e. they are not assigned their own predicates (Carroll et al., 1999) (cf. §4.1). The ERG/ACE realization relies on hand-written trigger rules, which signal that particular semantically empty lexical items may be required, but the set of rules does not always account for every scenario, which can lead to failed realization attempts.

2.4.2 Python libraries: pyDelphin and pydmrs

DMRS graphs can be manipulated using two focused Python libraries. The `pyDelphin`[26] is a more general *MRS-dedicated library. Throughout most of our work we used the 0.9 version to convert MRS structures produced by ACE into DMRS graphs and to interface with ACE from within Python code. The `pydmrs` library[27] (Copestake et al., 2016) is dedicated solely to DMRS manipulations.

All the code produced as part of this thesis is in Python 3.5.

2.4.3 Visualisation

An indispensable tool for working with DMRS is a visualiser. We heavily relied on Demophin[28], a DELPH-IN web demo, producing DMRS graphs directly from ACE output. The library `pydmrs` includes a tool by Matic Horvat, which adapts Demophin to display DMRS from XML. All the DMRS figures presented in this thesis were produced by one of these tools.

[26] https://github.com/delph-in/pydelphin
[27] https://github.com/delph-in/pydmrs
[28] https://github.com/goodmami/demophin or a version hosted on the University of Washington server: http://chimpanzee.ling.washington.edu/demophin/erg/.

Chapter 3

Rule-based DMRS chunking

In this thesis we explore the concept of a semantic chunk as a self-contained unit of meaning that can be processed independently without loss of useful information. In Chapter 2 we considered this definition in the context of existing semantic terminology. The remainder of the thesis focuses on practical aspects of semantic chunking for DMRS and surface string representations.

The experiments described in this chapter evaluate our earlier claim that syntactic clauses are suitable candidates for semantic chunks (§2.1.3). We start by investigating finite clauses with a subject-verb structure in a closed set of grammatical constructions (§3.1). Finite clauses express propositions and are the most self-contained type of clauses, since they can occur on their own as standalone sentences and are valid inputs of tasks such as parsing. At the same time, they are the most restrictive interpretation of semantic chunks (§2.1.2).

The first chunking method we propose acts as a proof of concept for the task. The system described below is based on a set of hand-crafted rules for detecting chunking opportunities in DMRS graphs. The rules cover a selection of grammatical constructions which introduce coordinated and subordinated finite clauses. In §3.5 we examine the produced chunks and then demonstrate how they can be applied to realization (§3.6). Although realization is the primary target task of our chunking system, we introduce parsing as an additional objective, which is possible because of the close relationship between the two operations within the *MRS framework (§2.4.1). The resulting semantic chunks become the basis of our Chapter 4 investigation into surface-based semantic chunking.

The prototype chunking system described in this chapter has its origins in the author's Master's dissertation, and the related preliminary results were published at the 2016 ACL student workshop as *Graph- and surface-level chunking* (Muszyńska, 2016). The realization experiments from §3.6 were published at the 10$^{\text{th}}$ International Conference on Natural Lan-

guage Generation (INLG '17) as *Realization of long sentences using chunking* (Muszyńska and Copestake, 2017).

3.1 Finite clause chunks

A key property of a semantic chunk is its independence from the rest of the sentence. We discussed in Chapter 2, in particular §2.1.4, how this constraint affects the choice of suitable candidates for chunks. The experiments in this chapter assume realization (§2.4.1) as the target application of semantic chunking. The chart generation used in the *MRS framework leverages the bidirectional nature of the ERG grammar and can be considered an inverse to parsing. The generator finds potential surface strings matching the input structure and attempts to match their grammar-based analyses to the original one. If the input MRS does not obey the grammar constraints, its realization fails.

Taking this into account, we choose to err on the side of strictness and maximally restrict the form of allowed DMRS chunks. A valid semantic chunk for the purpose of the rule-based prototype chunker is a subgraph of a DMRS representing a finite clause in a subject-verb form. Since finite clauses of this form can exist as standalone sentences, they can be straightforwardly processed by the generator, assuming they are sourced from a well-formed parse.

At the same time we choose the chunk form with the secondary objective in mind. Semantic chunks created in this chapter become the starting point of the experiments described in the next one. In particular, we use them as a training dataset for a semi-supervised surface chunking model (§4.3). Leveraging the bidirectional nature of the ERG, we chose parsing as the target task of the surface chunking, as the symmetry of the two processes suggests that they can benefit from similar chunks. The focus on the chunking precision ensures a high-quality dataset and consequently fewer errors propagating down the pipeline.

Finite clauses are represented in DMRS by semantic subgraphs centred around a tensed node (§2.3.4). Because of the way in which the ERG encodes copular constructions (§2.3.1), the node can be a verb, an adjective or a preposition. In this chapter we exclude from our consideration subjunctive and relative clauses. Subjunctive clauses, such as the subordinate clause in Example 53 (*(he should default on the loan = (if) he ever happens to default*), are often indistinguishable in their surface form from declarative clauses with a modal verb (= *it would be wise of him to default*). Without the disambiguating context of the rest of the sentence, the correct analysis cannot be guaranteed.

(53) Figaro promised Marcellina to marry her if he should default on the loan.

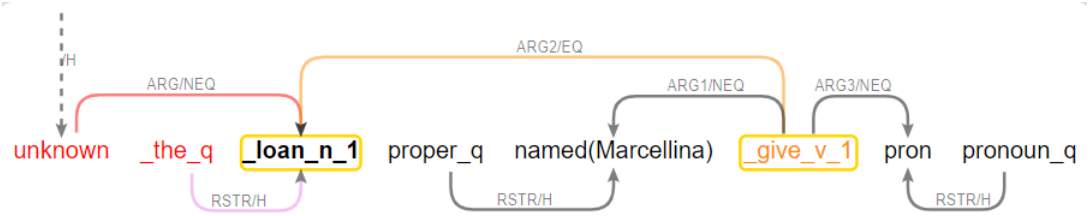


Fig. 3.1 DMRS of *the loan Marcellina gave him*.

Relative clauses also lie outside the scope of this chapter's investigation, despite being finite and having a well-defined subject. Their representation in DMRS is more complicated than the other types of finite clauses. First, the antecedent of the relative clause is shared between the two chunks, e.g. *the loan* in:

(54) Figaro defaulted on the loan Marcellina gave him.

As discussed in §2.1.4, elements shared between semantic subgraphs do not preclude their suitability as semantic chunks because the relevant information can be preserved during chunking. However, since the system we develop in this chapter is intended as a simple prototype model, we reserve the investigation of more complicated cases as a further research objective. Another complication stems from the fact that the top node of a semantic subgraph corresponding to a relative clause is the modified noun (`_loan_n_1` in Fig. 3.1) rather than an event node like in other chunk subgraphs. Finally, surface strings of relative clauses can be interjected into the string of the main clause, leading to non-consecutive surface representations of semantic chunks. Such fragments are not suitable to act as surface-based semantic chunks for the purpose of our experiments in Chapter 4, which are the secondary target task of our model.

Additionally, we make a special exception for sentences like *It changes as is required.*, where the subordinate clause is finite but lacks a subject.

3.2 Chunking rules

In order to arrive at the chunking rules, we considered a variety of grammatical constructions in English with regard to clause types that participate in them, using Huddleston and Pullum (2002) as reference. Among the structures with finite clauses, we identified the ones which can be unambiguously detected in DMRS graphs. The main three types of chunking decisions we focused on are associated with:

- clausal coordination,

- subordinating conjunctions,
- clausal complements of verbs.

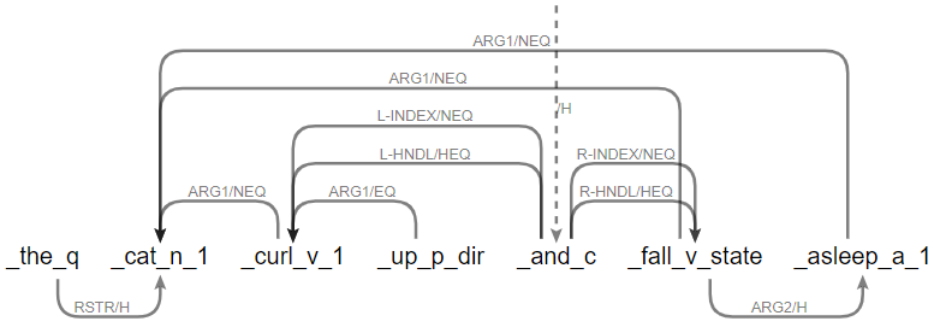
These constructions are associated with a fixed set of predicates and characteristic link structure. In the early stages of our research the notion of semantic subgraphs was not fully crystallised. We designed the rules with an intuitive notion of constituents in mind. As discussed in §2.1.4, there is noticeably more interaction between elements within a constituent than between elements from different constituents. The interaction in DMRS takes the form of links, so that there are many links between the nodes of a single chunk but only few crossing chunk boundaries. Leaning towards a strict interpretation of constraints, we require subgraphs of the chunks in the rule-based system to connect to the remainder of the DMRS via a single node, acting as a gatekeeper to the subgraph (with the exception of coordination when the top and top index of coordinates diverge, cf. §2.3.2).

The rules for detecting chunking opportunities define **trigger nodes** and **trigger links**, which indicate the presence of one of the selected constructions. Trigger links start or end at their corresponding trigger nodes and act as sole connectors between the trigger and the chunks it produces. Trigger nodes are all scopal operators and, in most cases, functional, closed-class words which do not belong to any chunk. The exception are verbs with clausal complements. For simplicity we avoid potential trigger nodes which participate in an undirected EQ link (§2.3.2).

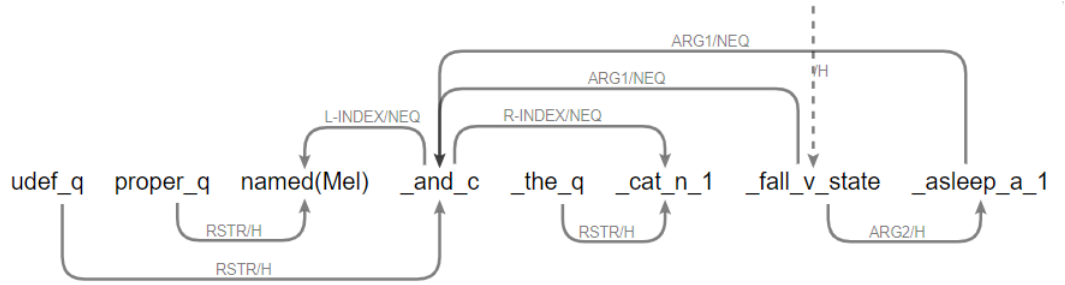
3.2.1 Clausal coordination

Let us consider clausal coordination as an example of how we arrived at the chunking rules. The trigger for coordination is the coordinator node, e.g. `_and_c`, taking the coordinates as its arguments via links with unique labels L-INDEX, R-INDEX, L-HNDL and R-HNDL. The INDEX links point to the top index nodes of coordinate subgraphs and the HNDL links lead to the coordinates' top nodes. Examples of DMRS graphs with coordination are presented in Figure 3.2.

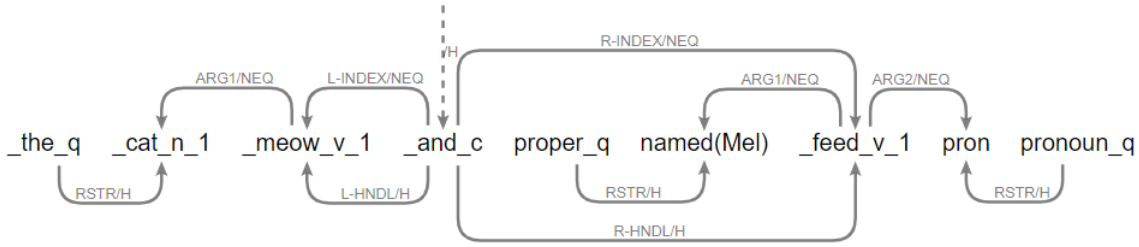
The link structures of the graphs reflect differences in the type of coordinates. Following the stipulation that only finite clauses are valid chunks, only the clausal coordination DMRS (Fig. 3.2c) can be used for chunking. VP coordination suffers from the same problem as relative clauses (§3.1) – the subject is shared between two chunks. For simplicity we do not consider such chunks in the current model. Comparing the link structures of the three scenarios, we can recognise that clausal coordination has unique link labels. All four links have to be present in the DMRS, unlike for the NP coordination, which does not have HNDL links. What distinguishes clausal coordination from VP coordination is the type of links: H



(a) DMRS with a VP coordination: *The cat curled up and fell asleep.*



(b) DMRS with a NP coordination: *Mel and the cat fell asleep.*



(c) DMRS with a clausal coordination: *The cat meowed and Mel fed it.*

Fig. 3.2 Different types of coordination.

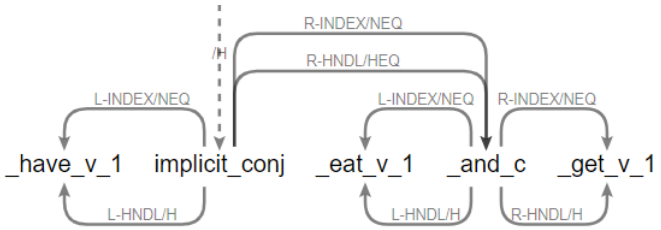


Fig. 3.3 Triple coordination in *Andy had ribs, Mel ate a doughnut, and the cat got nothing.* Only relevant nodes shown.

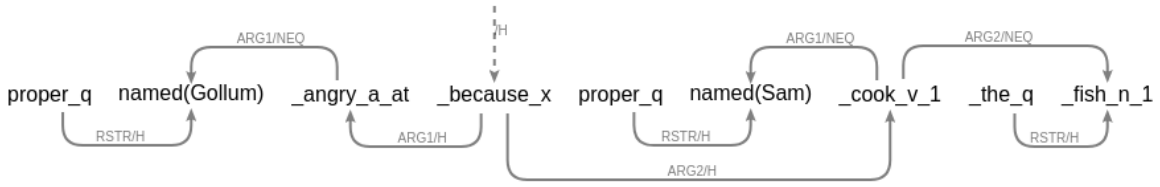


Fig. 3.4 An example of a DMRS with a subordinating conjunction: *Gollum was angry because Sam cooked the fish.*

as opposed to HEQ. The trigger node is any real-predicate node that is a source of the links matching the required trigger links.

Asyndetic coordination in the ERG 1214 has a unique grammar predicate `implicit_conj`, but mostly follows the same rules for the purpose of DMRS chunking. A special pattern appears in multiple coordination, where the rightmost coordination is the right coordinate of the implicit conjunction, but with a HEQ link instead of the usual scopal H (Fig. 3.3).

3.2.2 Subordinating conjunction

The second type of triggers we consider is subordinating conjunction, with the conjunction itself, e.g. *because*, acting as the trigger node. Based on Huddleston and Pullum (2002, Chapter 11), we composed a fixed list of predicates which appear with subordinate finite clauses. Subordinating conjunctions in the ERG are scopal operators with two arguments, and the scopal links act as trigger links for the chunks. The ARG1/H and ARG2/H links lead to the main clause and the subordinate clause respectively. An example of the construction in a DMRS is shown in Figure 3.4.

3.2.3 Clausal complements

The final type of triggers are verbs with clausal complements. Finite clauses which act as complements of other parts of speech can be valid semantic chunks as well, but for practical reasons we limit our preliminary investigation to verb complements. In particular, noun complements suffer from similar problems as relative clauses (cf. 3.1) when it comes to their surface representation: the complement string is interjected into the string of the chunk based on the main clause.

The other types of triggers are all scopal operators with functional predicates and they do not inherently belong to the chunks they introduce (§2.1.4). On the other hand, the trigger for clausal complement chunking is the node corresponding to the verb taking the clausal complement. It has an open-class predicate and is usually the central node (§2.3.3) of the main

clause subgraph. Its ARG2 or ARG3 argument introduces a subgraph of the complement clause through a scopal link ARG2/H or ARG3/H, with the exact form dependent on the presence of a direct object.

Although complement clauses can represent different moods, we only consider chunks formed from declarative complement clauses. Surface chunks based on interrogative clauses are not well-behaved for parsing, which is the secondary target application of our chunking system, as their correct analysis in the absence of the rest of the sentence cannot be guaranteed (§3.1). For example, the only difference between the DMRS subgraphs representing clausal complements in Examples 55 (declarative) and 56 (closed interrogative) is the sentence force property (cf. Table 2.2) of the verb *was*.

(55) Gandalf knew that Bilbo's ring was the One Ring.

(56) Gandalf wondered whether Bilbo's ring was the One Ring.

(57) Merry wondered what happened to the second breakfast.

When the clause *Bilbo's ring was the One Ring* is parsed, only the declarative analysis is recovered. The divergence is even more pronounced for open interrogative clauses (Example 57) as they share some features with relative clauses.

Although not explicitly marked as such in DMRS, some clausal complements are compulsory in the grammar. The ERG represents many verbs by the same predicate in all subcategorization scenarios, but sometimes different predicates are used depending on what arguments are present. For example, the verb *say* is parsed into a predicate `_say_v_1` in Example 58a, but into `_say_v_to` in Example 58b.

(58) a. She said.

b. She said that it was fine.

Once the subordinate clause is removed during chunking, a parser cannot assign the correct predicate to the verb based on the surface chunk of the main clause alone. This violates the meaning preservation condition of semantic chunking.

Subcategorization preferences of each predicate are encoded in the grammar's lexicon as lexical types (§2.3.1). When considering a predicate with a finite clausal complement as a chunking trigger, we check whether it appears in the lexicon with a lexical type that allows the parser to interpret the corresponding bare verb as the same predicate as the verb with a complement. By inspection, we found that no matter what other lexical types the predicate supports, it can exist in its bare form if one of its lexical types is `v_np*_le` or `v_pp*_le` (cf. §2.3.1). The asterisk indicates that such verbs can take optional NP or PP arguments. We report the results for chunking with and without the lexicon access in §3.5.

3.3 Chunk creation

In the previous section we gave an overview of the chunking rules designed to find semantic chunks corresponding to finite clauses. The chunks are created by the local application of the rules. As the first step we identify all potential trigger nodes within the DMRS. Most of them are scopal operators and can be organised in a hierarchy (§2.3.2). By extension, this relationship also holds between the resulting chunks. Since we identify triggers locally, one sentence can contain multiple unlinked trigger hierarchies, which can be, however, related to each other based on the positioning of the triggers inside chunks. For instance, Example 59 (simplified in Fig. 3.5) contains two triggers (in bold): `_after_x_h`, which is the top of the DMRS, and `_announce_v_to` in the relative clause.

(59) **After** they argued for a while, the Council was surprised by Frodo, who **announced** that he would take the Ring to Mordor.

There is no path made up of scopal links joining the two, but if we start the chunking process from the trigger closest to the top node of the DMRS, we can locate the other trigger inside the chunk produced by the first one. As a result, the two scopal hierarchies are separate but can be ordered relative to each other within the sentence.

We start chunking a sentence from the topmost trigger in the scopal ordering. Its trigger links lead to top nodes of semantic subgraphs for chunks lower in the hierarchy (cf. Fig. 3.5). As a consequence of our assumptions about the form of semantic chunks in this chapter (§3.1), the top node of a chunk subgraph is the subgraph's unique gatekeeper node. If it is removed, the subgraph becomes disjoint from the rest of the DMRS and so we can identify the DMRS elements belonging to the chunk by finding nodes that connect to the DMRS top node only through the gatekeeper[1]. The resulting subgraph of the new chunk is then in turn explored in search of lower trigger nodes, and so on, until we reach a chunk without a trigger.

A semantic subgraph is a valid chunk if a) it represents a suitable clause (§3.1), and b) it does not contain any nodes already accounted for by previously explored chunks. For example, the L-INDEX link in coordination in Figure 3.6 skips a level in the chunk hierarchy because of diverging top and top index nodes. It points to the same node as the *even if* trigger, violating the single gatekeeper criterion.

Most chunking decisions are associated with operators which do not belong to either of the created chunks. The chunking algorithm stores the operators in a dedicated **functional subgraph** (§2.1.4), together with any nodes not accounted for by chunk subgraphs, such as operators and modifiers acting on the trigger itself. A disconnected trigger subgraph indicates a chunking failure.

[1] We ignore coordination INDEX trigger links for coordination triggers.

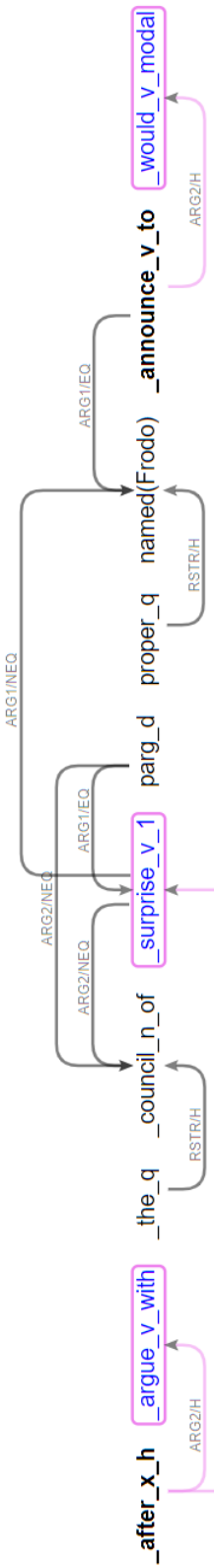


Fig. 3.5 Simplified DMRS of Example 59. The bold nodes are triggers and the framed ones are the tops of semantic chunks.

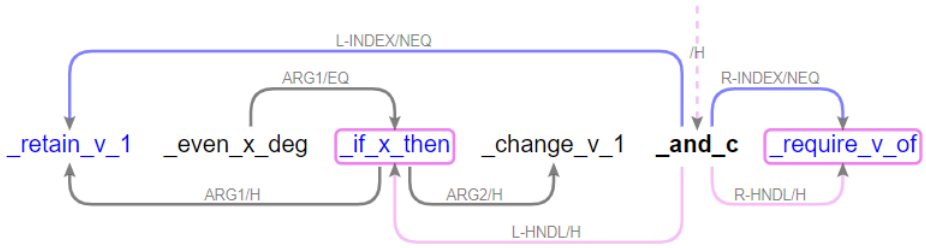


Fig. 3.6 Simplified DMRS for *It retains the structures even if circumstances change, and it requires no checks.*

3.4 Dataset: WeScience

The main dataset used throughout this thesis is WeScience (Ytrestøl et al., 2009), based on a selection of 100 NLP-themed articles from a 2008 snapshot of Wikipedia (Ytrestøl, 2009). The sentences in the corpus were treebanked as part of the Redwoods corpus (Oepen et al., 2004), which adapts the idea of treebanking to the needs of the constantly evolving ERG. Whenever a new version of the grammar is released, the corpus is manually validated and the updates are automatically propagated through the corpus to yield gold standard annotations.

The gold standard analyses are the reason why we choose the smaller WeScience over its larger parent Wikiwoods corpus (Flickinger et al., 2010) for the majority of the experiments in the thesis. We hope that the annotated data reduces errors associated with incorrect parsing that can propagate down the chunking pipeline. We use the version of the dataset parsed with the ERG 1214 and extracted from the LOGON[2] (Lønning et al., 2004) distribution of the Redwoods corpus. The MRS version was produced with the `redwoods` script available with the installation, and then converted to DMRS with `pydelphin`.

Because of its source in Wikipedia articles, the dataset contains many non-sentential entries. In the articles they introduce links, suggest related articles, and play other functional roles. We restricted the dataset to examples with DMRS graphs that:

- are fully connected,
- have a well-defined top node,
- have at least one tensed node.

We further edited the entries with the enhanced set of heuristics for determining target nodes of scopal links (§2.3.4). The resulting dataset comprises 6929 DMRS graphs.

[2]Website: http://www.emmtee.net/

	# chunked examples	% dataset
no-comp	1188	17.1%
all-comp	1401	20.2%
lex	1265	18.3%

Table 3.1 How many examples were chunked at least once?

	# occurrences		
Trigger type	no-comp	all-comp	lex
clausal coordination	677	657	672
subordinating conj.	591	574	585
clausal complement	—	310	109
Total	1268	1541	1366

Table 3.2 Number of triggers of each type.

3.5 Results

Below we present the results of semantic chunking of the WeScience dataset with the rule-based chunker described in the earlier sections of this chapter, as well as some dataset statistics. The examples were chunked in three different trigger configurations, varying in their treatment of clausal complements:

no-comp without clausal complements,

all-comp with all clausal complements allowed,

lex with clausal complements based on the grammar lexicon (cf. §3.2.3).

For example, the sentence in Example 60 will only be chunked in the second configuration. It will not be chunked in the third configuration because of the different lexical entries of the verb *say* (§3.2.3).

(60) Sam said that he didn't trust Gollum.

Table 3.1 shows the number of examples for which at least one chunking opportunity was found, while Table 3.2 shows the distribution of trigger types. Complement triggers added in the all-comp configuration are responsible for 21% of all chunking decisions, but only 35% of them are allowed in the lexicon-verified set up. At the same time, the addition of complement chunking increases the coverage across sentences only by 3%, indicating that complement chunks rarely appear in simple sentences without other types of covered constructions. The counts of subordinating and coordination triggers change once the complement chunks are included because of interaction between the two trigger types.

The average number of nodes in a DMRS from the dataset is 26, and the largest graph has 111 nodes. Among the examples where at least one chunking opportunity was found, the average rises to 34 nodes, compared with 24 for unchunked sentences[3]. As expected, longer sentences are the main target of chunking because they usually have more complex structure.

For example, our chunking system finds four semantic chunks in the sentence in Example 61, accompanied by three functional subgraphs. The string fragments corresponding to the chunks are marked with square brackets.

(61) If [the articles are omitted, which is sometimes done in headlines ("Mann beißt Hund"),] [the syntax applies as in English] — [the first noun is the subject] and [the noun following the predicate is the object.]

The size of the DMRS makes it difficult to show it in a figure. The three triggers are: the initial *if*, the implicit coordination indicated in the string by the dash, and the *and* coordinator.

For comparison, the largest unchunked DMRS has 79 nodes and corresponds to the sentence:

(62) Recreational amenities are scattered throughout the campus and include a workout room with weights and rowing machines, locker rooms, washers and dryers, a massage room, assorted video games, Foosball, a baby grand piano, a pool table, and ping pong.

Since the length of the sentence stems mostly from a simple NP enumeration rather than a complex grammatical structure, it falls outside of the domain of semantic chunking.

Relative chunk sizes

Ideally, a chunking operation divides the sentence representation into equal parts. If one chunk is considerably smaller than the other, the complexity cost of processing the larger chunk is not affected sufficiently to justify chunking. Coordination and subordinating triggers result in a distribution of chunks centred about roughly the 50% mark (coordination: av. 48%; subordinating: av. 53%), although the subordinating distribution is slightly skewed towards the main clause (Fig. 3.7). Chunks produced by complement triggers are more uneven – on average 70% nodes belong to the complement clause. This is due to examples with main clauses such as:

(63) She stated (that…)

(64) It has been argued (that…)

[3]The numbers in this paragraph are quoted for the all-comp configuration, but differences between set-ups are negligible.

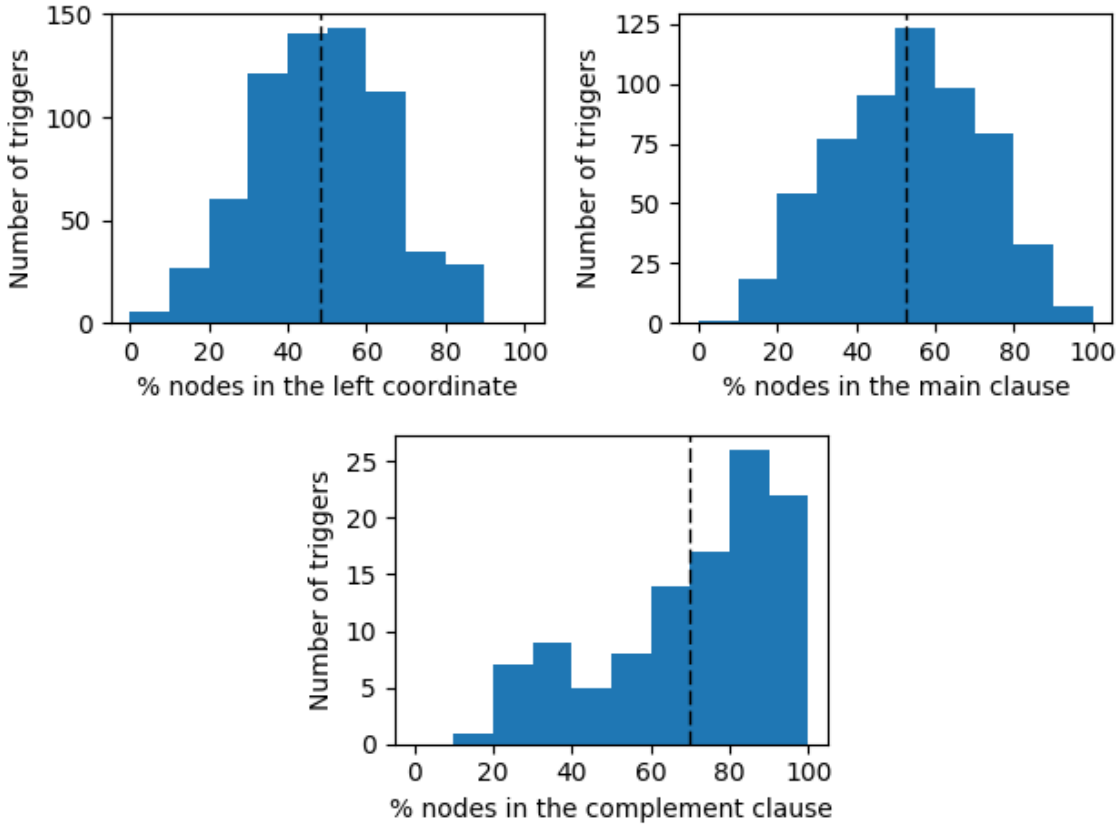


Fig. 3.7 Relative sizes of chunks for each trigger type (lex). The node count includes all the nodes in the chunk subtree (§3.3) introduced by each trigger.

Even though the resulting chunks represent situations and satisfy theoretical aspects of the Chapter 2 discussion, they are questionably useful from the practical standpoint. In both examples the main clause chunk corresponds to a single DMRS node. The chunker would benefit from a balancing check, which would prevent chunking when the trigger yields significantly uneven subgraphs or if the subgraph under consideration is already small.

Size of functional subgraphs

Semantic chunks are typically introduced by a trigger node representing a function word. With the exception of triggers for complement chunks, trigger nodes do not belong to any content chunk, but act as connectors in the hierarchy outlined by operators. A trigger, together with all the connected nodes unassigned to a chunk, forms a functional subgraph (§3.3). Since functional graphs have to be accounted for separately in the processing, optimal chunking would minimize their size, so that the actual semantic chunks contain a vast majority of the semantics of the sentence. 90% of functional chunks for coordination contain only the trigger node, compared with 79% of subordinating functional chunks. For instance, all functional subgraphs in Example 61 comprise just one node.

In practice, subordinating prepositions are often modified, e.g. *even though* or *just as*, increasing the size of the functional subgraph. The largest functional subgraphs, however, emerge in the presence of top-level sentential modifiers, such as the initial modifiers in the following examples:

(65) [Given a query]$_F$, [the system translates it]$_{C1}$ [and]$_F$ [the retrieval module finds matching documents]$_{C2}$.

(66) [For example,]$_F$ [it is silent in one word,]$_{C1}$ [but]]$_F$ [we pronounce it in another.]]$_{C2}$

In the first sentence the nodes corresponding to *given a query* belong to the same functional subgraph as the trigger *and*. Any processing pipeline making use of chunks needs to incorporate the information stored in the functional graphs, especially in cases where the functional fragments modify the semantics of chunks rather than act as connectors between them. We propose some approaches in our realization experiments in Sections 3.6 and 5.4.

Error analysis revealed that our chunking system does not correctly address coordination of clausal complements in examples where the token with the complement is not recognised as a trigger. For example, the following sentence contains an interrogative clausal complement of a noun:

(67) There are frequent debates among Esperanto speakers about [whether a particular borrowing is justified] or [whether the need can be met by deriving from existing words].

The bracketed fragments correspond to subgraphs identified as semantic chunks in the DMRS of the sentence. The complement clause as a whole is not a valid semantic chunk according to our chunking rules, because nouns are not valid triggers (§3.2.3). At the same time, the complement itself is a clausal coordination and each coordinate is a valid chunk based on that construction. The main clause contains the DMRS top, and the complement clause is headed by the *or* trigger. As a result, the chunker assigns the entire main clause to the functional subgraph associated with the trigger. We could restrict chunking to operators with the direct scopal connection to the DMRS top, but that would further restrict even non-problematic chunking opportunities. Another potential solution would be for the chunker to check that the functional graph does not represent a clause. This difficulty with the treatment of rarer scenarios and unforeseen interactions between grammatical structures highlights the limitations of a system based on a locally-checked small set of rules.

3.6 Realization with chunked DMRS

Semantic chunking is a practical pre-processing task with no standalone purpose. In order to test our chunks, we need a target task. In this section we evaluate semantic chunking and its associated processing paradigm on the example of realization.

Surface realization, or **generation**, is a task of producing sentences from representations. It can be thought of as the inverse of parsing. Constraint-based approaches implemented, for instance, using chart algorithms make it possible to use the same reversible grammar for both parsing and realization, as in the case of the ERG and its processors (§2.4.1). There is a variant of the task definition which includes deciding on the content of the output, e.g. in dialogue systems, but in our research we focus on realization from existing representations. Because of the close relationship with parsing, realization from *MRS is sensitive to the well-formedness of its input, making it a suitable objective to challenge the high-precision chunking system we designed.

Large semantic representations, usually corresponding to long sentences, present a challenge to generators. In the worst-case scenario computational cost of chart generation increases exponentially with the complexity of representation, although the algorithm can be modified to improve the practical performance (Carroll et al., 1999; White, 2004). Below we demonstrate how the efficient realization of a full sentence is possible through a principled composition of realizations of its chunks. The technique noticeably reduces the cost of realization and in some cases, allows for realization where no result was found using the standard approach. This effect is achieved without significantly degrading realization quality.

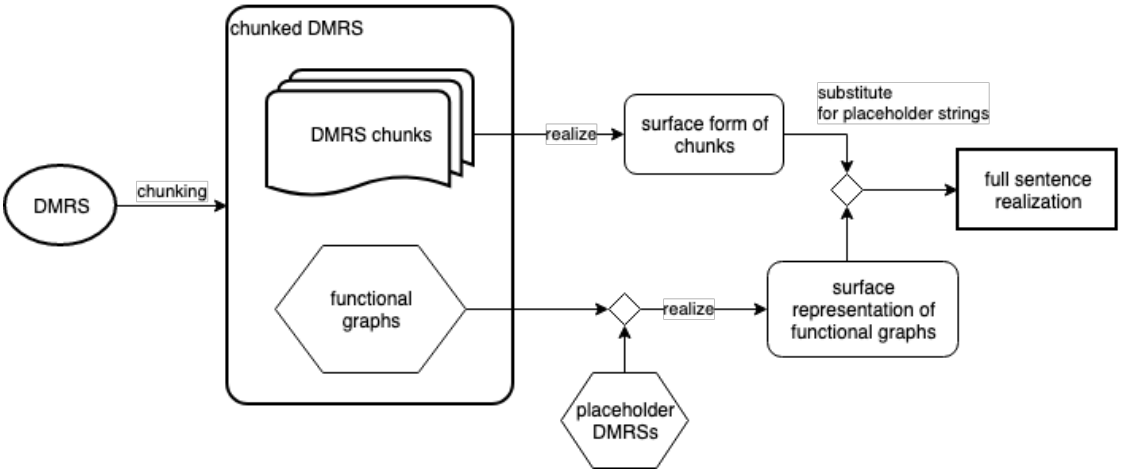


Fig. 3.8 Realization with chunks

Realization from chunks occurs in four phases. After chunking a sentence, some chunks, and in particular the functional subgraphs, do not meet the well-formedness criteria required for successful generation. As the first step we make sure that each subgraph is a well-formed DMRS, i.e. it has a well defined top node and meets grammar constraints. If needed, we introduce small placeholder graphs to account for compulsory arguments (§3.6.1). During the main phase each chunk is realized separately, and finally, the chunk realizations are combined with the processed functional fragments into a full sentence realization, based on how the DMRS chunks were originally linked. Figure 3.8 schematically illustrates the whole process.

3.6.1 Placeholders

The information about how chunks interact is stored in functional subgraphs (§3.3). They comprise trigger nodes (with the exception of the complement trigger) and any modifiers scoping over multiple chunks. Since they do not represent complete constituents, functional subgraphs are not well-formed DMRSs and fail any processing attempts. For example, the simplest functional subgraph for a clausal coordination is made up of just the coordinator node, without its compulsory coordinate arguments. In order to represent the content of functional subgraphs in a surface form, we need to convert them into a valid generator input. We achieve that by enhancing them with small pre-defined graphs, **placeholders**, which occupy places at the end of trigger links where we would originally find semantic chunks.

The two minimalistic placeholders we use correspond to clauses *it was snowing* and *it was raining*. Their DMRS graphs consist of single nodes and have one possible realization, which

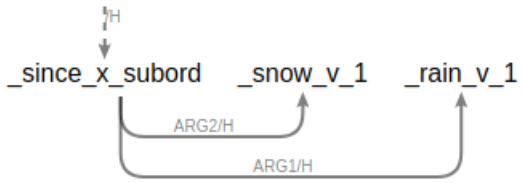


Fig. 3.9 DMRS with placeholders: *Since it was snowing, it was raining.*

is important for the assembly phase. They also do not occur naturally in the Wikipedia-based dataset we use for our experiments (§3.4). The resulting DMRS for the functional graph based on a subordinating conjunction *since* would consist of three nodes: `_since_x_subord`, `_snow_v_1` at the end of the highlighted ARG1 link and `_rain_v_1` at the end of the ARG2 link (Fig. 3.9).

Chunking based on a clausal complement relies on a trigger which is part of the semantic chunk of the main clause. The only information stored in the functional subgraph keeps track of trigger links severed in the chunking process, but the subgraph has no nodes. In order to obtain the surface representation of the connection between main and complement clauses, we enhance the main clause chunk with a placeholder in the same way as for the other triggers. After all chunks are realized, the known surfaces of placeholders can be replaced with surface representations of appropriate chunks and the full sentence can be assembled recursively based on the chunking decisions.

3.6.2 Example

Let us walk through the entire process for the following example (Fig. 3.10):

(68) [The Fellowship left for Mordor]$_{C1}$ **after** [the Council **had decided** that Frodo would carry the Ring.]$_{C2}$

The chunking algorithm finds two triggers in the sentence: subordinating conjunction with `_after_x_h` and a clausal complement with `_decide_v_1`. The latter trigger is a scopal argument of the former and so they can be straightforwardly ordered in the scopal hierarchy. The trigger links for subordinating conjunction lead cleanly to two tops of semantic subgraphs which represent main and subordinate clauses. They form two new semantic chunks, equivalent to C1 and C2 bracketing, with the functional subgraph comprising a single conjunction node.

The C1 chunk subgraph (Fig. 3.11a) does not contain further triggers, but the C2 subgraph includes a previously found trigger for the clausal complement chunk: `_decide_v_1`. Its associated trigger link leads to the top of the complement clause subgraph (`_would_v_modal`,

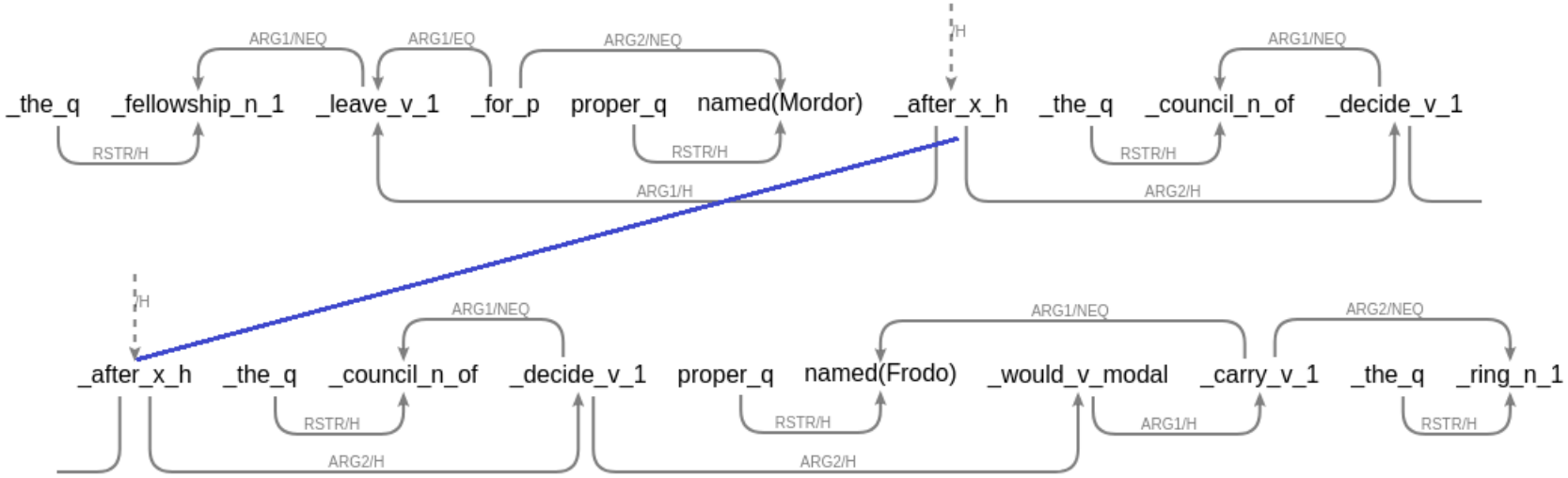
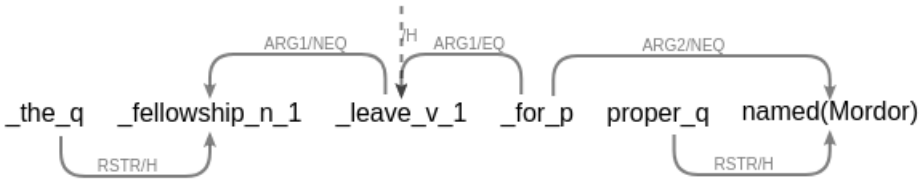
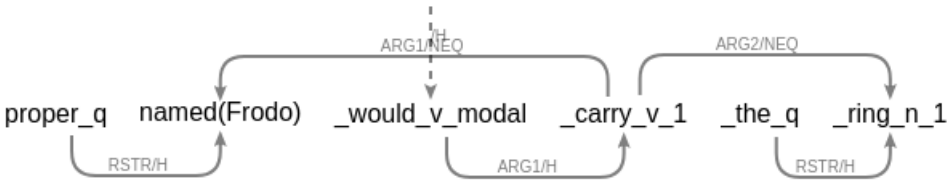


Fig. 3.10 DMRS of Example 68. The graph was split in two parts to fit on the page. The blue line indicates the node alignment.



(a) Semantic chunk, C1, equivalent to *the fellowship left for Mordor.*



(b) Semantic chunk equivalent to *Frodo would carry the Ring.*

Fig. 3.11 Semantic chunks from Example 68.

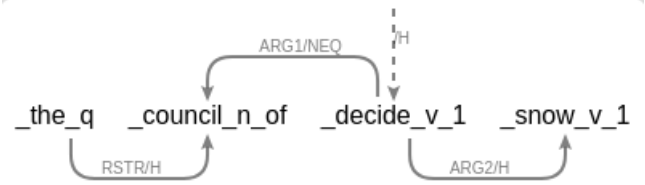


Fig. 3.12 Semantic chunk equivalent to *the Council had decided*, enhanced by a placeholder.

Fig. 3.11b) and satisfies the single gatekeeper requirement. Separating the subgraph requires only the removal of the trigger link, so that the corresponding functional graph contains no nodes. The two final chunks without further triggers realize as:

(69) the fellowship left for Mordor

(70) Frodo would carry the ring

The third chunk corresponding to the *the Council had decided* contains a trigger and before it is input to the generator, we enhance it with a placeholder graph in lieu of the complement (Fig. 3.12) in order to establish the syntactic relationship between the main clause and the complement. The augmented realization results in:

(71) the council had decided that it was snowing

Similarly, we generate from the enhanced functional graph of the subordinating conjunction to obtain two syntactic variants:

(72) after it was raining it was snowing

(73) it was snowing after it was raining

Both variants are correct, even though only one matches the original. We discuss the ranking of chunked results in the next section, together with issues such as punctuation.

We know that *it was snowing* and *it was raining* are placeholder surface strings, which can be recursively replaced by actual chunk realizations. For the first syntactic variant of the subordination:

(74) after (it was raining → the council had decided that it was snowing)
(it was snowing → the fellowship left for Mordor).

(75) after the council had decided that (it was snowing → Frodo would carry the ring) the fellowship left for Mordor

(76) after the council had decided that Frodo would carry the ring the fellowship left for Mordor

3.6.3 Experiment

The following investigation was originally published as a short paper at the 10th International Conference on Natural Language Generation (INLG) as *Realization of long sentences using chunking* (Muszyńska and Copestake, 2017). The experiments described in this section were conducted with an earlier version of the rule-based chunker than the one described in previous sections. They correspond to the `lex` configuration of the chunking, where the lexicon information is used to determine whether a complement clause of a verb should form a separate chunk (§3.5). The differences in the system stem mostly from changes in its implementation and the development of DMRS-dedicated algorithms. As such, we decided that there was no need for the reprisal of the experiments because of their proof-of-concept nature and conclusive results. We are confident that any disparities in the results would be in favour of the newer system if the experiments were to be repeated.

We compare the results of realization from a chunked sentence with the standard full-sentence realization. The standard set-up uses default ACE settings with two adjustments: a) a fixed time-out of 30 seconds after which a realization attempt is abandoned; b) an added step for dealing with unknown words (§2.4.1). The time-out chosen is quite high and does not affect most realizations.

Semantic chunking primarily aims to benefit long sentences. To investigate them specifically, we filtered 315 sentences from the WeScience dataset which have DMRSs with more than 40 nodes and which can be chunked at least once[4]. There are on average 3.6 chunks per sentence (max. 12).

[4]The sentences were not pre-processed with the additional scope-resolving heuristics (§2.3.4).

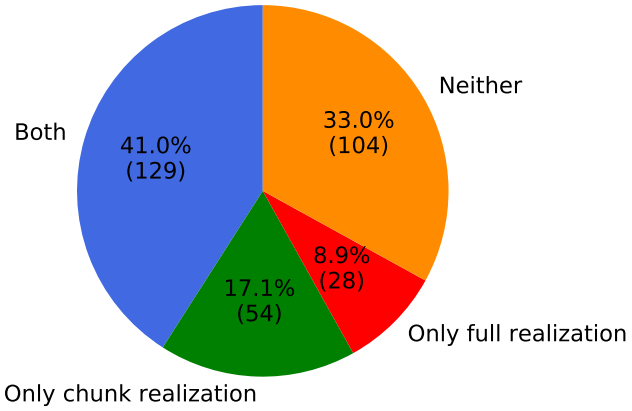
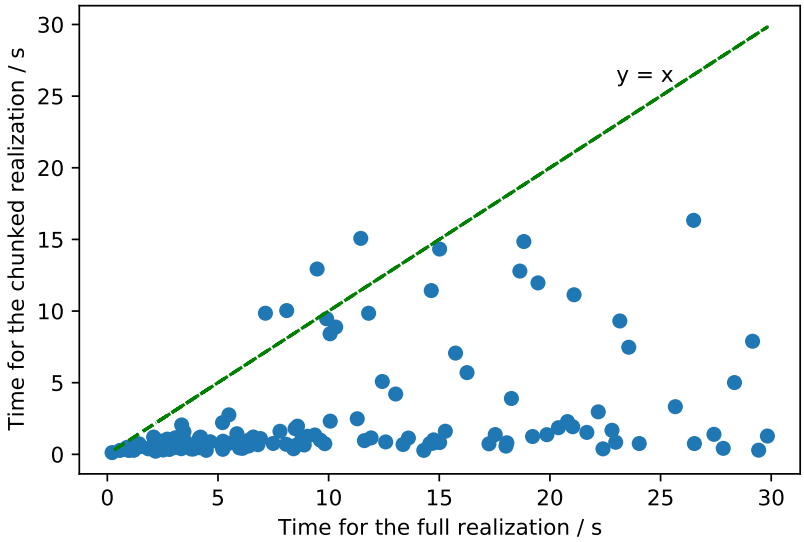


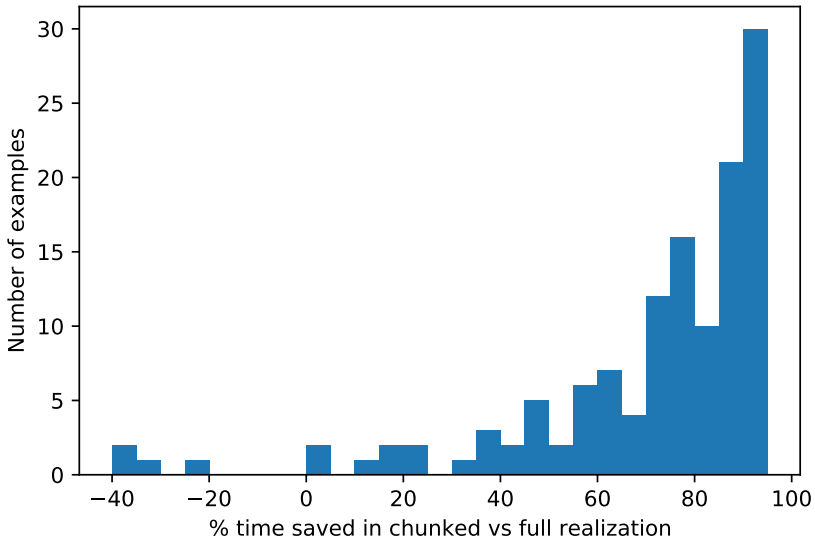
Fig. 3.13 The number of examples for which the standard realization and/or realization with chunking were successful.

Realization with chunking allowed generation from some semantic graphs which do not produce a sentence in the standard set-up. While the standard approach resulted in a successful realization for 157 examples (49.9%), realization with chunking worked 183 times (58%), yielding coverage that is about 8% higher (Fig. 3.13). Some sentences cannot be realized with the new method even though the standard system works. We attribute these cases to chunking errors, such as the result of the interaction between complements and coordination (§3.5).

We investigated the performance of the two approaches in terms of time and memory usage. Figures 3.14 and 3.15 show CPU time and RAM used, as reported by ACE, for all examples where sentences were successfully realized with both methods or where the standard realization failed without a time-out. The time measured for realization with chunking was the sum across all chunks; the maximum time was 16 seconds. On average, we observed a 73% reduction in the time required for realization when chunking was used, and a 63% reduction in the RAM needed. The outliers in the upper half of the scatter graph and on the left-hand side of the bar chart correspond to sentences where the standard realization failed or chunking was incorrect; this accounts for around 3% of all examples. The shape of graphs for maximum memory used matches that for time, including the outliers. An alternative measure of the complexity of realization is the number of intermediate structures (passive edges) produced in chart generation. It also follows a similar pattern and shows improvement consistent with that observed in the RAM and time measurements, including the occurrence of outliers.

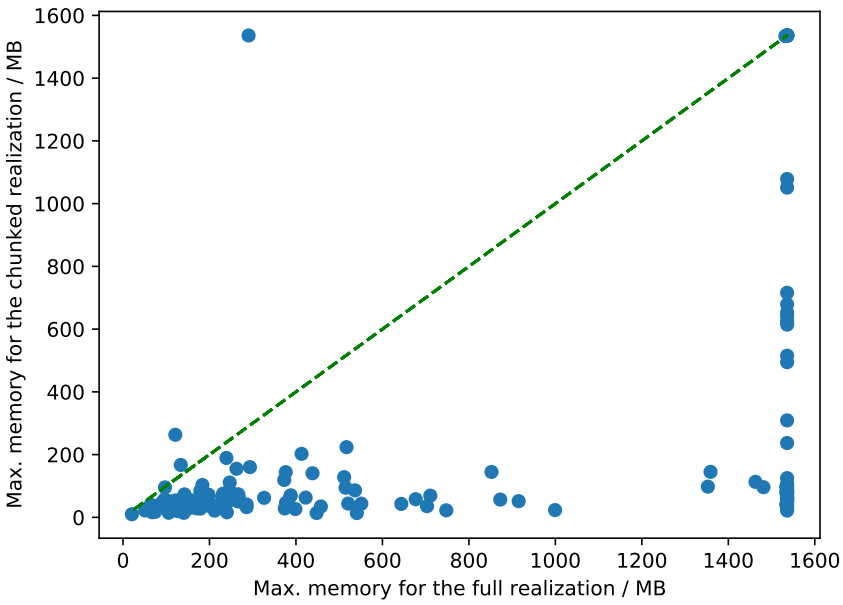


(a) Time needed for realization with chunking against time taken by the standard realization.

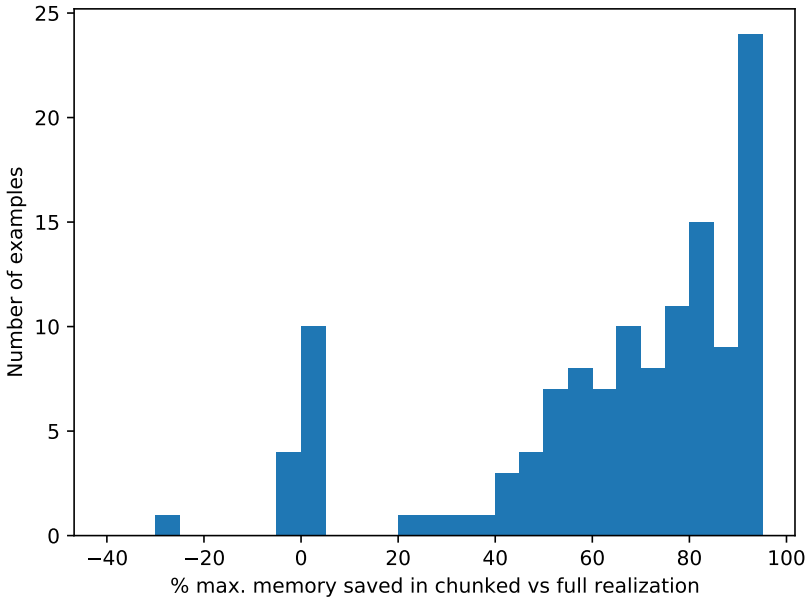


(b) Time saved by the introduction of semantic chunking.

Fig. 3.14 Realization with chunking reduces time used for realization by average 73% (median 81%).



(a) RAM needed for realization with chunking against RAM used up by the standard realization.



(b) RAM saved by the introduction of semantic chunking.

Fig. 3.15 Realization with chunking reduces RAM used for realization by average 63% (median 77%).

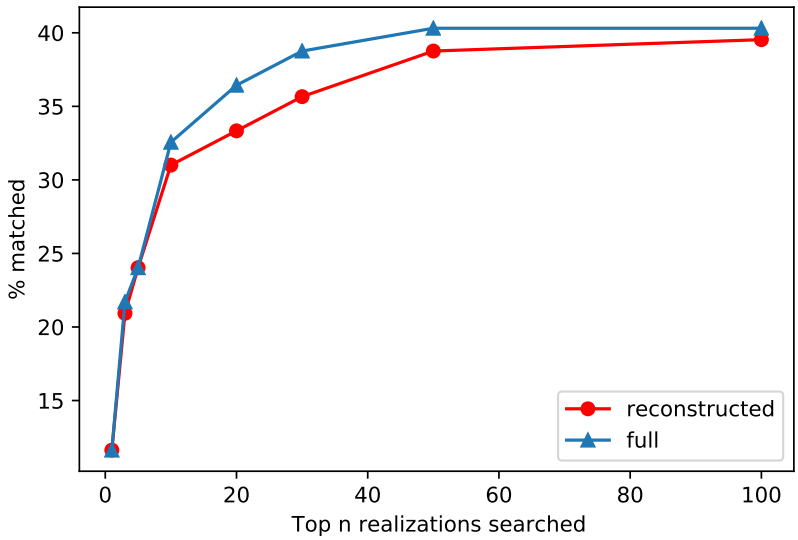


Fig. 3.16 Percentage of exact matches in top n realizations.

Apart from reducing the resources needed by ACE, realization with chunking has the potential to affect the ranking of generation results. The ACE generator ranks each surface form and assigns it a score, using a maximum entropy language model. In order to similarly score a sentence realized with chunking, we use the sum of logarithms of scores of the constituent chunk realizations. Following the original work on the ERG generator by Velldal (2008), we evaluate the ranking quality by comparing the top-ranked realization result with the original sentence on which the semantic representation was based, using three metrics: the exact match percentage and the BLEU score (Papineni et al., 2002) and the Levenshtein distance (Levenshtein, 1966).

We report an exact match in top n realizations of the examples realized by both methods, if the original and realized surface strings are identical after removing capitalization and punctuation. As Figure 3.16 shows, realization with chunking yields comparable results, especially for smaller n. In fact the differences at all levels are not statistically significant ($p > 0.1$). The data takes a form of paired observations of successes and failures for the two approaches under consideration. An observation is a success when the original string is present in the given top n outputs, and we can compare the competing outcomes for each sample. Taking this into account, we evaluated the significance of our results with a two-tailed sign test. This non-parametric test is a special case of a binomial test with the probability of success equal to 50% in the null-hypothesis scenario.[5].

[5]All significance tests were performed using the `scipy` package (https://scipy.org/)

The BLEU score (Papineni et al., 2002) is a metric commonly used in machine translation that captures similarity between a source and target text based on n-gram precision. In our experiment the BLEU score is evaluated only for the top ranked realizations, again after removing punctuation and capitalization. The average score for the standard realization is 0.79, and 0.77 for realization with chunking, including the examples realized by only one of the methods and using the original sentence as the gold standard. When we consider only sentences realized by both approaches, the standard approach achieved a higher score for 17.1% examples, while realization with chunking scored higher for 12.4%.

Similarly as for exact matches, the data consists of paired observations, but this time we can determine numerical differences between the two sets of samples, i.e. between the scores for the two approaches. To accommodate the change in circumstances, we assess the significance of these results using a two-tailed Wilcoxon signed-rank test (Wilcoxon, 1945). It is a commonly used non-parametric paired-difference test that compares population means of two set of observations. The null hypothesis states that the observations come from the same distribution and that, as a result, the distribution of differences between them is symmetric about zero. Our alternative two-tailed hypothesis is that one of the methods performs better than the other, i.e. the difference is not symmetric. The test indicates that the difference between the mean BLEU scores reported in the previous paragraph is not statistically significant ($p > 0.1$).

An alternative measure of text similarity is the Levenshtein distance, defined in terms of string operations and related to the minimum number of single-character edits needed to match two strings. It is zero when the strings are equal, and it cannot be less than the difference of their lengths. For the sentences realized with both methods, the average values of the Levenhstein distance from the original sentence are 49 for the reconstructed variant and 45 for the standard approach. The difference is not statistically significant[6].

Based on our experiments, chunking for realization may lead to a small deterioration in the quality of produced realizations. The difference in the BLEU scores of the two approaches is small (and not statistically significant). In the light of the considerable reduction in the processing costs achieved by the proposed technique, this is quite promising. If replicated in other experiments, a small difference of a similar order of magnitude as the one in our experiment might well be an acceptable trade-off for the gains in other areas. Furthermore, our implementation of realization with chunking relies on a simple method for the ranking of its results. A dedicated ranking model would improve the quality of top realizations. Since realizing semantic chunks individually narrows down the search space of the generator, the modified implementation has potential to surpass even the standard approach, as it might

[6]Two-tailed Wilcoxon signed-rank test (Wilcoxon, 1945), $p > 0.05$.

limit spurious analyses that incorrectly identify constituents and interactions between them. The process of retraining the ranking model is, however, not straightforward and remains a topic for a potential future investigation.

3.7 Conclusions

In this chapter we started to explore practically the task of semantic chunking introduced in theoretical terms in the previous chapter. We showed how a complex semantic representation like DMRS can be divided into semantic chunks, using a prototype chunking model based on a set of hand-crafted rules (§3.2). The rules define trigger nodes and links, which reliably indicate a presence of one of the selected grammatical constructions: clausal coordination, subordination and clausal complements of verbs. They were chosen because they lead to chunks corresponding to finite clauses with a subject-verb structure. Each chunking decision is associated with a functional subgraph, which stores information about inter-chunk connections.

We chose the constraints on the form of chunks with two objectives in mind: a) the task of realization, b) DMRS-string conversion of chunks for parsing. After investigating the produced chunks and their distribution, we applied them to realization, using a divide-and-conquer paradigm. Although generating surface forms of semantic chunks themselves is straightforward, realization with chunking has to account for the information encoded in functional subgraphs, necessary for the reconstruction of the full surface representation from partial results. In §3.6 we suggested the use of small placeholder DMRS as one possible solution to obtaining the surface representation of the auxiliary structures. The results described in §3.6.3 show considerable improvements in the resource requirements of realizing sentences with large DMRS representations. Realization with chunking takes on average 73% less time and 63% less memory than standard realization. Apart from the reduction in the memory and CPU time needed for the generation, chunking also sometimes allows for creation of a surface form when the standard approach fails. Furthermore, we observed no statistically significant difference between the quality of the realization with chunking and of the full realization.

In the next chapter we focus on the secondary objective of the chunks produced by the prototype model – creation of a dataset of surface chunks, based on DMRS subgraph chunks. We investigate how well suited the resulting string fragments are to act as semantic chunks for the task of parsing, and we train a string-based sequence labelling model, which allows us to apply chunking to tasks taking string input, such as parsing.

Chapter 4

Surface chunking

Semantic chunking is defined on a per representation basis. We assume from the outset that different representations might require different definition of what constitutes a valid chunk. The primary representation discussed in our work is DMRS, but we cannot ignore the fact that the majority of systems tackling natural language tasks operate directly on surface strings of sentences. It is unrealistic for us to require full semantic parsing as a pre-processing step to semantic chunking, itself intended as a pre-processing measure. Semantic chunking of more complex representations remains a sensible approach for tasks which already take such representations as input or use them internally.

At the same time, bare string representation does not expose the information in a way that is readily accessible to automated processing. This poses a dilemma: on one hand, the output of many tasks can be used to enhance the information available to processors; on the other hand, the tasks themselves often require an extra level of processing. As we discussed in Chapter 1, dividing the surface form of sentences into semantic chunks is not straightforward. Creating a set of ad hoc rules dedicated to a particular task that can produce a non-trivial annotation on a string level is prohibitively complicated and unreliable, requiring considerable effort and the involvement of an expert annotator. Such rules have to handle ambiguity, potential non-grammaticality and flexible punctuation standards, among other things. In Chapter 1 we brought up examples such as coordination and adverbial sentence modifiers to illustrate how simple string- and even syntax-level heuristics fail to capture the necessary level of nuance. As a result, the rule-based approach has been largely abandoned by the NLP community in favour of machine learning algorithms. In this chapter we frame surface-based semantic chunking as a sequence labelling problem, which allows us to leverage insights from established tasks such as PoS tagging, named entity recognition (NER) and shallow chunking (§2.2).

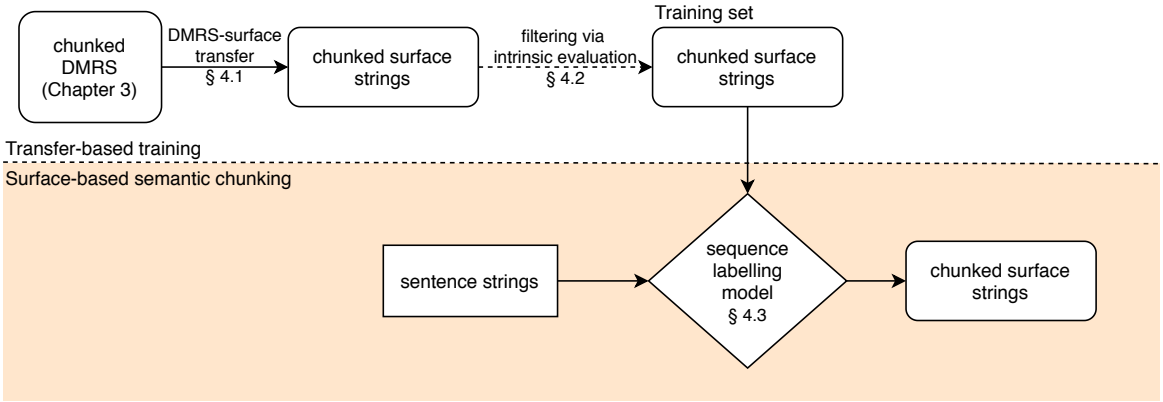


Fig. 4.1 Surface semantic chunking.

Supervised machine learning algorithms require large amounts of annotated data, usually expensive and tedious to obtain. A fixed annotation scheme makes it impossible to tailor the form of semantic chunks to the target task at hand, as each target application would require a separate annotation effort. In this chapter we explore an intermediate route of a semi-supervised approach. The direct link between a DMRS and a surface string in the form of character spans associated with each node (§2.3.1) allows us to transfer semantic chunks between the two representations. In this way we can use the DMRS-based chunks created in Chapter 3 to obtain the equivalent surface fragments that are capable of acting as surface semantic chunks.

The chunking system from Chapter 3 was designed with this purpose as its secondary objective. The intended application of the surface chunks is parsing, chosen because of the close relationship of realization and parsing in the *MRS framework (§2.4.1). Furthermore, the two tasks are well-suited to the semi-supervised approach we take in this chapter, as their inherent connection to the surface form allows us to move fluidly between the two representation.

For the proof of concept experiments described in this chapter (Fig. 4.1) we convert the chunks from Chapter 3 into their surface equivalents (§4.1) and evaluate to what extent the translated pieces behave as semantic chunks for string representations of sentences (§4.2). Finally, we frame surface semantic chunking as a sequence labelling problem and train a model capturing the underlying semantic decisions. In our experiments we are guided by successful solutions to similarly framed tasks (§4.3).

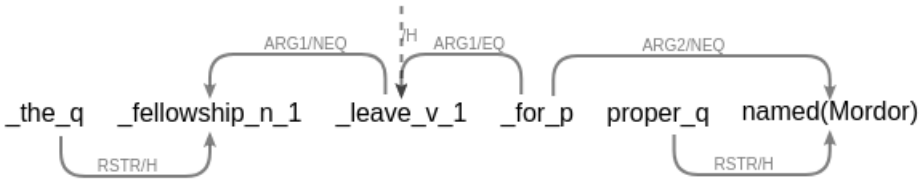


Fig. 4.2 DMRS chunk corresponding to *the Fellowhip left for Mordor*.

4.1 Subgraph chunks to surface chunks

In order to train a surface chunking model, we need a large dataset of chunked sentence surfaces. In this section we investigate how the surface chunks can be obtained by transferring DMRS-based chunks into their surface versions.

DMRS chunking splits DMRS graphs of sentences into chunk subgraphs connected by functional graphs, with every node assigned uniquely to one of the components. We use the term **component** to encompass both semantic chunks and functional fragments. In *MRS each predicate is associated with a character span in the source surface of the analysis (§2.3.1). The string covering the spans of nodes in a single component can be considered the surface representation of the component. We implicitly rely on this property throughout the thesis whenever we refer to a DMRS chunk by the surface string to which it corresponds. For example, in §3.6.2 we used the chunks in Example 77 for realization. Below we marked the surface parts that are equivalent to the DMRS chunks and functional fragments.

(77) [The Fellowship left for Mordor]$_{C1}$ [after]$_{F1}$ [the Council had decided]$_{C2}$ [that]$_{F2}$ Frodo would carry the Ring.]$_{C3}$

The first DMRS chunk (C1) is shown in Figure 4.2. Aligning nodes to tokens is mostly straightforward, but even in this simple case, one of the nodes (`proper_q`) has no surface equivalent. We explain the details of the alignment process in §4.1.1.

Surface semantic chunks based on DMRS chunks are intended to serve as training data for a sequence labelling model (§4.3). Because of that, we limit the form of surface components to consecutive spans of tokens in order to keep surface semantic chunking similar to related sequence tagging tasks. The consecutiveness assumption holds for most chunks, since semantic chunks are contained constituents of a representation and tend to behave as a unit (§2.1.4), but some functional subgraphs correspond to disjoint spans (cf. §3.5) and their associated chunking decisions cannot be used for surface chunking.

Subgraph chunks are created recursively. Large subgraphs are gradually divided into smaller and smaller ones, based on the presence of triggers (§3.3). Any chunking decision is optional, as we can decide not to separate two chunk candidates depending on considerations

such as their size. Based on this observation, we assume that two valid surface chunks created by one trigger are at most separated by the surface string of the trigger's functional graph. An overlap between spans of two components invalidates all chunking decisions that led to the separation of their subgraphs. As a result, not all chunking decisions made for subgraphs are transferred to surface chunks.

4.1.1 Alignment

Each component can be linked to a list of tokens. To achieve that, we can mostly follow the node alignment information. The mapping is straightforward for many content nodes. For example, the span of singular noun nodes simply corresponds to the lemma of a node (cf. Example 77). More complex grammatical properties, such as tense, introduce auxiliary tokens, while grammar nodes often do not introduce their own tokens at all (§2.3.1). As a result, some nodes correspond to multiple tokens and some surface spans are not explicitly linked to any nodes, e.g. infinitival *to* or some auxiliary verbs. They are often referred to as null semantic items, since they are strictly syntactic markers with no individual semantic meaning. In the ERG they are the subject of trigger rules responsible for their correct realization (cf. §2.4.1). Their treatment in *MRS is in line with the usual practice in formal semantics (Copestake et al., 2005). Horvat (2017) developed a set of heuristics for aligning the ambiguous spans and we use his code in our work. The alignment heuristics were designed for PTB tokenization, which differs from the usual ERG treatment. Following Horvat (2017), we use the spaCy tokenizer[1]. Generally, if a token is not aligned to any node but is surrounded by tokens already assigned to a component, we assume it also belongs there.

Some grammar predicates correspond to individual tokens, just like real predicates. These include predicates with `carg` values, like proper names, and pronouns. Many, however, require special treatment. Functional grammar predicates, e.g. `subord` used for non-adverbial clausal modifiers, are linked to token sequences already accounted for by other nodes. They are introduced directly by grammar rules at the later stages of parsing, rather than by the presence of corresponding lexical items. For example, the `subord` node introducing the modifier in Example 78 (Fig. 4.3) spans over the entire clause *annoyed by Sam*.

(78) Annoyed by Sam, Gollum threw a tantrum.

Because of this overlap, we include spans of grammar predicates into chunk spans if they align only with tokens which are otherwise unaccounted for. We found that edge tokens of the spans which partially overlap with other nodes are the most likely to be misaligned. If a

[1]https://spacy.io/

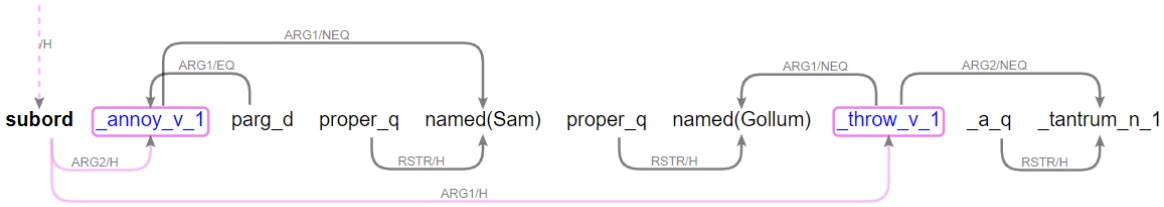


Fig. 4.3 DMRS of Example 78: *Annoyed by Sam, Gollum threw a tantrum.*

grammar predicate aligns with a span of tokens t_{ij}, i.e. from i^{th} to j^{th} token in the sequence, we add its tokens to the surface component with an already assigned span t_{mn} if $i < j < m$ or $n < i < j$.

Not all tokens get assigned to surface chunks at this stage. In particular, we still need to account for any punctuation tokens between chunks and other tokens not covered by alignment heuristics. It could be argued that they belong to functional fragments, since in Example 79 the comma would not be there if not for the subordinating construction:

(79) As the hobbits walked through a mushroom field, they got hungry.

To maintain the consecutive spans, we assign punctuation marks to components based on their relative order in the surface. In the beginning of the sentence, the span of the first chunk is extended to include them. Similarly, the last chunk includes any unaligned tokens from the end of the sentence. If an unaligned token falls between two chunks, we assign it to the preceding fragment, unless it is based on a functional graph. The exceptions are punctuation tokens which always fall at the end of a phrase, e.g. commas.

Some components do not have a unique surface equivalent, e.g. functional graphs for asyndetic coordination are made of a single `implicit_conj` node aligned with the entire span of the coordination (often the entire sentence). In the string, it is indicated by punctuation, sometimes including capitalization, but nothing in the DMRS itself points to how the coordination is represented in the surface string.

The one-to-one correspondence between DMRS and surface chunks breaks down also because of complementizers for chunks based on clausal complements. They are semantically empty function words and are not represented by nodes, but play a similar role to operators, such as *if*. When chunking the surface of a sentence based on a clausal complement, we assign complementizers to separate functional fragments. When used as a complementizer, a *that* token does not naturally align with any node. It can be handled by the alignment heuristics, but is not reliably assigned to correct nodes and it is not even clear what the correct node is. The first intuitive choice is the verb with the complement argument, which is the source of complementizer. On the other hand, complementizers are more commonly considered parts of the complement clause. Clauses such as 80 are ambiguous: the top two

Chunked examples	1265 (17.1% of the dataset)
Surface chunked	1117 (88.3% of the chunked)
Failed triggers	159 (11.3% of all triggers)

Table 4.1 Results of the DMRS-surface chunk transfer.

DMRS analyses disagree on whether *that* is a complementizer or if it belongs to a noun phrase *that Bilbo*.

(80) that Bilbo's ring was the One Ring

Before we apply the alignment heuristics, we exclude from consideration all complementizer tokens unassigned to nodes based on the DMRS information. If a matching token falls between chunks but precedes a chunk based on a clausal complement, we treat it as a standalone functional fragment (cf. Example 77). Otherwise, it is assigned to a chunk like any other leftover token in the final round of alignment.

4.1.2 Experiment

Below we present the results of the DMRS-surface chunk transfer. In particular, we investigate how well the chunks produced in Chapter 3 can be represented as consecutive strings. We evaluate their quality and suitability for the task of parsing in §4.2.

The starting point for our experiments is the result of the rule-based chunking described in Chapter 3, where we chunked DMRS graphs of 6929 sentences from the WeScience 1214 dataset (Ytrestøl et al. (2009); §3.4), using three different configurations (cf. §3.5). In the current experiments we use the version in which semantic chunks based on clausal complements are allowed only if the verb of the main clause is analysed with the same lexicon entry with and without the complement clause present (§3.2.3).

Many DMRS in the dataset can be chunked at multiple points, but the transfer procedure ignores the chunking opportunities which result in disjoint surfaces. For example, a graph in which we identified five DMRS chunks can yield a surface with just three chunks. Only 12% of examples which were chunked in their DMRS form cannot be separated into suitable chunk-based surface fragments (Table 4.1), as 11% of triggers cannot be used for string chunking (Table 4.2).

The most common reasons behind conversion failure are interactions between different types of triggers. In particular, a coordination in a complement clause (Example 81) leads to mischunking if the complement was not separated from the main clause during chunking.

(81) [Murphy suspected that]$_F$ the [Red Court was involved]$_{C1}$ [and]$_F$ [she should call Harry]$_{C2}$.

Trigger type	Failed
Coordination	83
Subordination	70
Clausal complement	6
Total	159

Table 4.2 Triggers which failed the transfer. They amount to 11.6% of all triggers.

We already found this scenario problematic in realization with chunking (§3.6.3). Just like there, the issue stems from the fact that the nucleus clause of the sentence (*Murphy suspected that*) is assigned to the functional fragment with the coordinator *and*. The corresponding surface fragment is disjoint, invalidating the transfer.

Another major, yet familiar, reason for failure was the presence of initial sentential modifiers, such as *however* in Example 82.

(82) However, the program is unable to pass the Turing test, as even the casual user will often expose its mechanistic aspects in short conversations.

The functional fragment consists of nodes for the trigger *as* and its modifying adverb *however* which together form a disjoint component. Under the condition that surface chunks have to be consecutive string spans, the failure in the conversion in both of the above scenarios is expected.

4.2 Intrinsic evaluation

The surface fragments created in the previous section are candidates for the role of semantic chunks of the surface representation. When we evaluated DMRS chunks in Chapter 3, we used extrinsic evaluation on the downstream task of realization. Although semantic chunking is a practical task and chunks have to match the needs of their target application, the chunked examples created in the previous sections are intended as a training set for a sequence labelling neural network model. The form of the model makes it difficult to reliably investigate the quality of surface chunks using extrinsic evaluation alone. Its performance depends on many factors and hyperparameters, and the training is resource-intensive. The expensive feedback loop means that we are interested in maximising the quality of the input data. Instead of refining the chunks by iterating the entire training, we design an intrinsic evaluation that ensures a good starting point for the semi-supervised approach.

The chunks transferred from the DMRS subgraphs are designed to be finite clauses, which allows them to be sufficiently independent of each other (§3.1). In particular, at this stage of

our investigation, we are interested in whether the surface chunks we produced satisfy this condition for the target task of parsing. The evaluation procedure takes the following form:

1. chunk the DMRS (§3.3);
2. convert the DMRS chunks into surface chunks (§4.1);
3. parse the resulting surface fragments corresponding to semantic chunks;
4. match the DMRS from Step 3 against the original full DMRS (§4.2.1);
5. score the match (§4.2.2)

The results of the evaluation guide the creation of a semi-supervised training dataset for surface semantic chunking (§4.3.3) dedicated to parsing. Step 3 simulates the behaviour of the created surface fragments as if they were used in parsing with chunking, in a similar way that we used DMRS chunks for realization with chunking (§3.6). We ignore the functional fragments as incomplete phrases which cannot be parsed directly.

4.2.1 DMRS matching

In order to evaluate how closely the partial analysis provided by parsing a surface chunk fits the original DMRS, we need to compare the parsed graph with the original DMRS, in particular with the corresponding DMRS chunk on which the surface fragment was based (§4.1). The problem of finding similarity between graphs is called **graph matching**. We are not interested only in perfect matches, but would like to evaluate the degree of similarity between the chunk analysis and the best matching subgraph. To quantify the degree of similarity between graphs, we propose an F-score based measure (§4.2.2).

The inexact graph matching problem of the sort we seemingly face has been proved NP-hard (Bengoetxea, 2002), but we can avoid the high complexity thanks to the surface alignment information included with each parse, which allows us to simplify the problem of partial graph matching to that of **subsequence matching**. The reframed task is to find all the longest common subsequences between sorted sequences of nodes. We choose to sort the nodes first by their associated starting character indices, and then by the string representation of their predicates[2].

The link structure of a DMRS is determined by the subcategorization properties of its nodes, encoded as predicate-specific information in the grammar's lexicon (§2.3.1). This suggests that nodes should receive preferential treatment in the matching algorithm.

[2]For grammar predicates with `carg` arguments, this includes the `carg` value.

Furthermore, some nodes are pre-determined by other nodes, e.g. grammar quantifiers `udef_q` and `pronoun_q`. Before matching two subgraphs we remove nodes with following predicates: `pronoun_q`, `number_q`, `idiom_q_i`, `proper_q`, `udef_q`, `def_explicit_q`, `def_implicit_q`, `_a_q`, `_the_q`. Two nodes are equal for the purpose of matching if they have the same predicate and properties (§2.3.1).

Subsequence matching has an established dynamic programming solution. The first step collects the information on the length of subsequences m and n of lengths $|m|$ and $|n|$ in a table L, so that $L[i][j]$ stores the length of the common subsequence up to and including the i^{th} element of n and j^{th} element of m (COMMONSUBSEQUENCELENGTHS in Algorithm 1). The second step backtracks through the table, finding the longest subsequences (LONGESTCOMMONSUBSEQUENCES in Algorithm 1). The time complexity of the algorithm is $O(|n| \times |m|)$.

Algorithm 1 Find the longest shared subsequences of nodes.

Input: sequence n of length $|n|$, sequence m of length $|m|$

Output: the dynamic table L with $|m|$ columns and $|n|$ rows s.t. $L[x][y]$ is the length of the longest common subsequence between $n[:x]$ and $m[:y]$ (including the x^{th} and y^{th} elements

procedure COMMONSUBSEQUENCELENGTHS(n, m)
  if not m or not n **then return** []
  initialize L with 0s
  for $1 \leq i \leq |n|$ **do**
    for $1 \leq j \leq |m|$ **do**
      if $n[i-1] == m[j-1]$ **then**
        $L[i][j] \leftarrow L[i-1][j-1]+1$
      else
        $L[i][j] \leftarrow max(L[i][j-1], L[i-1][j])$
  return L

Output: List of sets of pairs of matching nodes (x,y), s.t. $x \in n, y \in m$, forming the longest common subsequence.

procedure LONGESTCOMMONSUBSEQUENCES(L, n, m, i, j)
  if $i == -1$ or $j == -1$ **then return** $[\varnothing]$
  else if $n[i-1] == m[j-1]$ **then**
    result $\leftarrow$ LONGESTCOMMONSUBSEQUENCES(L, n, m, $i-1$, $j-1$)
    for $s \in$ *result* **do**
      $s : \cap\{(n[i-1], m[j-1])\}$.
  else
    if $L[i][j-1] \geq L[i-1][j]$ **then**
      result $\leftarrow$ LONGESTCOMMONSUBSEQUENCES(L, n, m, i, $j-1$)
    if $L[i-1][j] \geq L[i][j-1]$ **then**
      result $\leftarrow$ LONGESTCOMMONSUBSEQUENCES(L, n, m, $i-1$, j)
  return *result*

Parsed chunks	2226 (92.1%)
Perfect score (1.0)	1508 (67.7%)
Av. score ±std.	0.93 ± 0.16
Av. imperfect score ±std.	0.77 ± 0.20

Table 4.3 Summary of the results of intrinsic evaluation.

4.2.2 Scoring

When scoring a match, we compare the original DMRS chunk with a subgraph of the parse of the surface chunk, including only the nodes participating in the longest subsequence and any links between them. The measure we propose is an adapted F-score measure, similar to the Smatch metric by Cai and Knight (2013). Precision and recall are calculated over nodes and links of the two subgraphs, but since links are highly dependent on nodes (cf. §4.2.1), we weigh their contribution by half.

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$Precision = \frac{\text{\# matching}}{\text{\# in the parse}}, \; Recall = \frac{\text{\# matching}}{\text{\# in the DMRS chunk}}$$

The first step in the evaluation procedure (§4.2) is parsing surface chunk candidates into DMRS. We use the ACE parser (§2.4) allowing both formal and informal analyses with fragments and setting a 30 second timeout, and convert MRS into DMRS with `pydelphin`. This results in 92.1% of strings successfully parsed. We investigated some of the problematic surface chunks and found that the failures were caused by issues such as foreign words, unusual mark-up, non-standard punctuation, or failed MRS-DMRS conversion.

Almost 68% of parsed surface fragments perfectly match the source DMRS chunk. The distribution of F-scores in the dataset is strongly skewed towards the high end, with the average score of 0.93 (Table 4.3). Figure 4.4 shows the part of the distribution corresponding to imperfect scores. Even the lower-scoring graphs maintain high quality on average. If we ignore the perfect matches, the average F-score reaches still reaches 0.77.

The chunks scored in the 0.9-1 range tend to be the top analyses according to the ranking model of ACE. Although they differ from the parses indicated by the treebank (cf. §3.4), the differences often correspond to a single link pointing to a different node, or an extra grammar node present. The examples with scores in the 0.8-0.9 range behave in a similar way, but the differences between the two DMRS analyses are more pronounced.

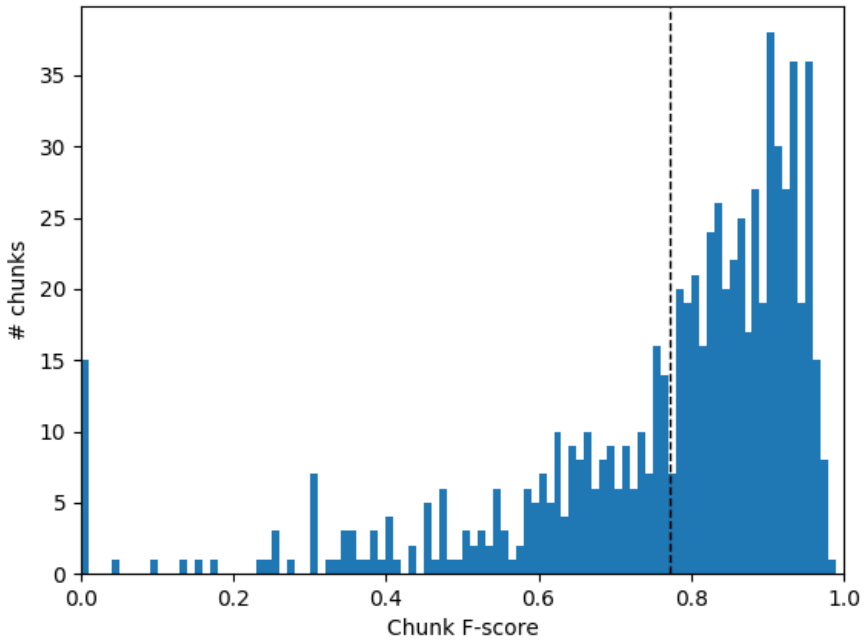


Fig. 4.4 The distribution of imperfect F-scores. The dotted line indicates the average.

The results of the intrinsic evaluation inform the creation of a training set for surface chunking. The WeScience dataset is too small to train a neural network model (cf. §4.3.3) and so we fall back on its non-treebanked parent corpus, WikiWoods, choosing the 0.8 score as the threshold for including an example in the training data. Although the models proposed in §4.3 are data-hungry, the strong average performance of transferred chunks in the intrinsic evaluation means that the chosen threshold provides plenty of training examples. We do not use a higher threshold in order to ensure a wide variety of represented chunks.The resulting dataset is presented in more detail in §4.3.3.

4.3 Chunk labelling

The first part of this chapter described how we created a surface chunk dataset by converting chunks of semantic representation into equivalent surface fragments. The conversion is possible thanks to the syntax-based nature of *MRS semantics and its strong compositionality, but it corresponds to an undesirable flow of information in most pipelines. Pre-processing operations on simple representations, such as strings, should not require access to complex representations, such as semantic parses. In this section we investigate whether chunking

decisions on surface strings can be successfully modelled directly, without access to the DMRS graphs of sentences.

A surface chunking model has to decide how to divide a surface string of a sentence into surface chunks and auxiliary functional fragments, given an input sequence of tokens and an annotated training dataset. The objective of partitioning a sequential input and classifying the resulting fragments is commonly framed as a **sequence labelling** problem, and applies to some of the most explored problems in NLP, such as part-of-speech tagging and named entity recognition (NER). Since these tasks typically target general linguistic properties of their inputs, they commonly serve as pre-processing steps for tasks which require deeper understanding of language and richer, more specific representation. It is one of many similarities between semantic chunking and some established tasks that we discussed in §2.2. Both shallow chunking and clause identification belong to the category of tasks commonly framed as sequence tagging, which suggests that the same approach can be successful for semantic chunking.

The input sentence is considered as a sequence of tokens $\mathbf{x}_{1:n}$, where n is the number of tokens in the sentence. The training objective is then to find the most likely sequence of tags, $\hat{\mathbf{t}}_{1:n}$, given the input sequence and the annotated training data, $\mathbf{X}$:

$$\hat{\mathbf{t}}_{1:n} = \arg\max_{\mathbf{t}_{1:n}} P(\mathbf{t}_{1:n}|\mathbf{x}_{1:n}, \mathbf{X})$$

.

The information gleaned from the training data is typically encoded in model parameters, θ, and the goal of the training is to find $\hat{\theta}$ that maximizes the ability of the model to find $\hat{\mathbf{t}_{1:n}}$ for a previously unseen sequence from the domain of the task.

In what follows we give a brief overview of models commonly used to address sequence labelling tasks. These state of the art approaches successful in similar applications guide our experiments described in §4.3.4.

4.3.1 Models

Neural machine learning models typically consist of encoding and decoding steps. An input sequence is first converted into a set of numerical features, weighted by the model architecture according to their predictive power on the training set. The decoding part of the model then selects the most probable tag for each token based on this internal representation. The independence of neural network architectures from expensive human feature design makes them a particularly appealing match to the flexible needs of semantic chunking. As a trade-off for reducing the design costs, neural models require large quantities of annotated

data. In the *MRS framework (§2.3) the annotation burden is shifted from runtime to the grammar creation, which can be considered a one-off intensive annotation effort (Bender et al., 2015). Thanks to the detailed grammar, ACE and other processors can produce *MRS analyses without further human supervision.

Assuming a large dataset is available, neural models can learn from it more efficiently. Our preliminary experiments involved constructing a feature-based MaxEnt classifier for semantic chunking, but computational costs of processing hundreds of thousands of sentences proved prohibitive, especially when paired with a slow model-design iteration associated with hand-engineered features. Training neural networks on the same dataset might take longer, but uses less time and RAM for models with larger computational capacity.

Unlike traditional models, neural architectures rely on distributed word representations, which assign a vector to each word, based on statistical properties of its occurrences in the text. Although it is possible to initialize the vector values randomly for each end-to-end task, there exist multiple pre-trained embedding variants, which can be used as priors, potentially leading to faster convergence and better results. They usually capture properties such as similarities between words and can be of particular use for semantically focused tasks. In our experiments we initialize the word vectors to GloVe embeddings (Pennington et al., 2014), following Ma and Hovy (2016) and Yang et al. (2018).

Recurrent neural networks (RNNs) are particularly well-suited to the domain of natural language sentences because their design allows the encoding of longer distance dependencies between input tokens. As such, they have dominated the sequence labelling tasks. Bidirectional models, such as bidirectional long-short term memory (BiLSTM; Hochreiter and Schmidhuber (1997), Gers et al. (1999), Dyer et al. (2015)), additionally condition encoding of each token on both its preceding and following contexts. Although recurrent cells capture some dependencies between inputs and it is possible to successfully project the model representation onto the label space with a softmax distribution (Ling et al., 2015), newer research shows that RNN models still benefit from a composition with a CRF decoding layer in tasks where there exist strong dependencies between consecutive labels (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016; Yang et al., 2018). This is the case for most sequence labelling tasks, as the most common tagging schemes follow an internal grammar. For example, in NER a B-ORG label marking a beginning of an organisation name cannot precede an I-PERS label reserved for tokens inside a person name. The same holds true for our labels (§4.3.2). The CRF models the joint probability of tags within a small window of the sequence based on transition probabilities observed in the training set, and the CRF Viterbi decoding algorithm ensures the best compromise of all constraints at the labelling time.

Current state-of-the-art approaches to sequence labelling tasks (Reimers and Gurevych (2017); Yang et al. (2018)) have three elements in common: 1) a layer creating character-level word embeddings, combined with pre-trained word embeddings; 2) a BiLSTM encoder; 3) a CRF tagger. Although the models we experiment with draw insights from the leading techniques, our goal is to demonstrate the viability of the sequence labelling approach in the context of semantic chunking, rather than the creation of a fine-tuned, best-possible model.

We do not expect semantic chunking based on our dataset to be difficult for models to learn compared with similar tasks, such as shallow chunking or NER, both commonly used as benchmarks for general labelling architectures. Surface chunks based on the prototype chunker from Chapter 3 cover a selected sample of grammatical phenomena which can be defined on the DMRS representation with a small set of simple rules. The complexity of the task is expected to increase for wider-coverage chunking schema, such as the system described in Chapter 5. Even then, since relationships between chunks can be encoded in structured templates, models should be able to represent them, given capacities typical of deep neural network architectures.

On the other hand, we expect the noise in the dataset. Additionally, the sentences in our dataset tend to contain only a few chunks each, which means that most tokens are labelled as belonging inside a chunk (cf. §4.3.2). As a result, accuracy is nearly useless as a metric and we focus on F-score instead, together with precision and recall. Since the interesting labels are separated by long token spans, our model relies strongly on the capability of BiLSTM to capture long-distance dependencies and the CRF to enforce the internal grammar of the tags.

4.3.2 Labels

Two of the most common tagging schemes used for sequence labelling are called BIO, and BIOES. The labels B, I and O respectively mark the beginning, inside and outside of segments of interest, which for sentence chunking are chunks and functional fragments. The BIOES scheme introduces additional tags for end (E) tokens and a special singleton (S) symbol for segments comprising a single token. The labelling schemes are interconvertible. The BIOES scheme is also known as BILOU with L (last) and U (unit) tags used in place of E and S. Ratinov and Roth (2009) and Yang et al. (2018) report the BILOU system to perform significantly better on the NER task than BIO, while Lample et al. (2016) and Reimers and Gurevych (2017) do not find a significant difference in the performance between the two schemes.

A BIO-scheme label comprises two parts, typically separated with a hyphen: B, I or O marking the place of a token in the segment, followed by a tag identifying the type of the segment. For example, in the NER task a B-ORG tag marks the start of a segment

Trigger type	Labels
Coordination	FCONJ, CCONJ
Subordination	FSUB, CSUB
Complement	MTHAT (main clause), FTHAT (complementizers), CTHAT (complement clause)

Table 4.4 Labels for chunks produced by different trigger types, with an initial C for chunks and an F for functional fragments.

corresponding to an organisation name, while I-PER labels tokens inside a person name. In our labelling scheme we recognise two major categories of fragments: chunks and functional fragments. They are tagged with C and F respectively.

Although our labels are based on the BIO tag scheme, the outside (O) label is never used, as every token belongs to either a chunk or a functional fragment. Let us consider the chunked sentence in Example 83, which contains two neighbouring functional fragments.

(83) [The hobbits had breakfast,]$_{C1}$ [but]$_{F1}$ [since]$_{F2}$ [they were used to elevenses,]$_{C2}$ [they got hungry quickly.]$_{C3}$

If we treated all functional fragments as members of the general outside category with the O label, we could not distinguish that the coordination *but* and subordinating conjunction *since* belong to two distinct functional fragments. This information is needed for the correct reconstruction of the full analysis from the results for individual chunks. To allow for this, we forgo the O label and treat functional fragments as another category of spans with its own beginning and end symbols. As a result, our task resembles sentence segmentation, but the similarity is not useful due to the unique reliance of sentence segmentation on abbreviations (§2.2.4).

When we obtain surface chunks from their DMRS counterparts (§4.1), we preserve the information about what type of trigger was responsible for each chunking decision (§3.2). This information is conveyed in the label. For example, tokens which belong to a chunk based on a coordination are labelled with tags containing CONJ. We introduce distinct labels for chunks based on main and complement clauses (MTHAT and CTHAT) as the main clause chunk is the only chunk type that contains a chunking trigger (§3.2.3). All segment types used are listed in Table 4.4).

Example 84 repeats the sentence from Example 83 but with bracketed spans converted to appropriate labels:

(84) The\B-CCONJ hobbits \I-CCONJ had\I-CCONJ breakfast\I-CCONJ ,\I-CCONJ but\B-FCONJ since\B-FSUB they\B-CSUB were\I-CSUB used\I-CSUB to\I-CSUB

elevenses\I-CSUB ,\I-CSUB they\B-CSUB got\I-CSUB hungry\I-CSUB quickly\I-CSUB.\I-CSUB

Since we do not represent nested relationships between different triggers, the CCONJ labels of the right coordinate of the *but* coordination are overridden by the SUB labels of the subordinating conjunction inside the coordinate. The role of the corresponding fragment as the right coordinate can be deduced from the label pattern and the assumption that chunks originating in one trigger are at most separated by that trigger's functional fragment (§4.1). Not all nesting decisions can be recovered in this way. In particular, it is impossible to order nested triggers of the same type. Exploring more complex labelling schemes that are capable of capturing nested relationships is a potential direction for future research (§6.2.1).

We evaluate the results of surface semantic chunking with a span-based F1 metric, for which the labels are converted to labelled spans and a successful tagging of a single component requires a correct start and end of the chunk span and a correct type, e.g. CCONJ.

4.3.3 Dataset: Wikiwoods

Neural machine learning models require substantial amounts of training data to successfully generalise over a task. For example, training portion of the CoNLL 2003 named entity recognition (NER) dataset comprises more than 200,000 tokens spread over almost 15,000 sentences (Tjong Kim Sang and De Meulder, 2003). The CoNLL 2001 chunking dataset includes almost 9000 sentences in its training section, divided among 107,000 chunks (Yang et al., 2018). The WeScience dataset we relied on so far is too small for this purpose (§3.4). Instead, we fall back to its "parent" dataset, Wikiwoods (Flickinger et al., 2010) — a larger, non-treebanked snapshot of Wikipedia. The version of the dataset parsed with the ERG 1214 has not been officially released at the point of writing, so we are working with an unofficial parsed data we created ourselves.

Out of all the Wikiwoods examples from the 1212 version of the dataset we selected the ones with 40 or more nodes in their DMRS in order to focus on the performance of chunking for examples where its application would be the most desirable. We filtered out wiki-specific entries and examples with no tensed nodes (cf. §3.4). The corresponding sentences were then parsed with ACE[3], and the MRS were converted into DMRS using the `pydelphin` library (§2.4.2).

Not all selected sentences yielded successful analyses, failing at parsing or the MRS-DMRS conversion. The successfully parsed ones underwent the surface chunking and intrinsic evaluation procedure outlined in §4.1 and §4.2. Based on the WeScience experiments,

[3]Informal root settings, enabled fragments, max. chart MB=12000, max. unpack MB 12000.

	Training	Validation	Test
Sentences	884,149	110,519	110,519
Chunks	1,912,644	239,248	238,939
Functional fragments	679,392	84,615	84,927
Tokens	34,378,084	4,295,590	4,292,440

Table 4.5 Properties of the Wikiwoods-based dataset for one of the random dataset splits.

Trigger type	Training	Validation	Test
Coordination (B-FCONJ)	359,448	44,593	44,734
Subordination (B-SUB)	271,142	33,765	33,950
Complement (B-MTHAT)	69,163	8,569	8,611

Table 4.6 Distribution of trigger labels for one of the random training-validation-test splits. The numbers do not include asyndetic coordination, which does not have an FCONJ fragment.

we decided to keep all examples with more than one surface chunk for which all surface chunks obtained an F-score higher than 0.8 (§4.2.2). The threshold generally corresponds to chunked analyses equivalent to the top result returned by ACE. It maintains a variety of chunking triggers (Table 4.6) and provides a sufficient number of examples to train a neural network model (Table 4.5). In fact, our dataset considerably outsizes existing datasets for related tasks mentioned in the beginning of the section. As part of our experiments, we train the model using a 10% portion of the training data (§4.3.4) in order to better place semantic chunking in the context of other sequence tagging tasks.

4.3.4 Results

We implemented all our models using the AllenNLP library (Gardner et al., 2018), built on top of the PyTorch deep learning framework (Paszke et al., 2019). The training proceeded with the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001. Apart from providing a relatively quick convergence on our task, Adam is among optimizers that show lower dependence on hyperparameters and on random seed values (Reimers and Gurevych, 2017). Once we had found a value resulting in a satisfactory convergence rate, we did not further fine-tune the hyperparameters. Although the models on which we based our system perform better with the standard stochastic gradient descent (SGD), the optimizer parameters which were successful for their training did not perform well on our task. Fixed learning-rate SGD did not reach a better final score and took much longer to converge. Adding the momentum and learning rate decay, like in the model by Ma and Hovy (2016),

almost completely stopped the progress of the task before the convergence was reached, although it is possible that the hyperparameter search could improve the score.

The baseline for our experiments is a BiLSTM with a 50-dimensional GloVe uncased embedding (Pennington et al., 2014), 200 hidden units and a softmax projection layer[4]. We enhanced the model gradually, first by replacing the softmax projection layer with a CRF tagger (`bilstm-crf`), then by enhancing the token representation with a concatenated character-level CNN embedding (`bilstm-crf-cnn`). Each character is represented by a 16-dimensional embedding and passed through 128 trigram filters, with ReLU activations (Nair and Hinton, 2010) and the hyperparameters values taken from Yang et al. (2018). We used a dropout of 0.5 in all configurations.

We repeated each experiment three times with different random data splits and random seed initializations[5]. Since semantic chunks found by the DMRS chunker correspond to long text fragments, the majority of tokens were marked with one of the inner chunk tags, e.g. I-CSUBJ, leaving the other labels comparatively underrepresented in the dataset. As such, instead of accuracy, we report span-based F-score values for all configurations averaged over the three trials in Table 4.7, together with their standard deviations. Its value on the validation set was also used as the stopping criterion for all models.

As shown in Table 4.7, our biLSTM baseline reached the F-score of 0.74, which indicates that sequence labelling is a good choice of framing the problem. Adding a CRF decoding layer (`bilstm-crf`) led to a major improvement ($\Delta = 0.12$), also noticeable in qualitative terms, as we observed a reduction in the number of examples for which the model failed to follow the internal grammar of the labelling scheme (cf. §4.3.1). The character-level embeddings (`bilstm-crf-cnn`) raised the score by a margin comparable with the observed standard deviation, which is not statistically significant. The scores for our models are generally lower than those reported by Yang and Zhang (2018) for shallow chunking and NER, which reach around 0.95 and 0.91 respectively. These are both well-established tasks and their state-of-the-art models have been fine-tuned over years of iterative progress. Our best score of 0.862 is a good result, considering the early state of research and the lack of in-depth fine-tuning.

As mentioned in §4.3.2, there is evidence from other tasks that the BIOES tagging scheme can give better results that the BIO variant. Although we did not observe a significant difference in the score (`bilstm-crf-cnn-bioes` vs `bilstm-crf-cnn`), the standard deviation of the BIOES scores was smaller, suggesting a better stability of the training. More trials would be required to confirm the effect.

[4] AllenNLP: Simple Tagger

[5] The three seeds were the same for all configurations.

Configuration	F-score
bilstm	0.737 ± 0.008
bilstm-crf	0.860 ± 0.018
bilstm-crf-cnn	0.862 ± 0.010
bilstm-crf-cnn-bioes	0.862 ± 0.001
bilstm-crf-cnn-random	0.857 ± 0.001
bilstm-small	0.674 ± 0.002
bilstm-crf-cnn-small	0.812 ± 0.002

Table 4.7 Results of surface-based semantic chunking.

In general, increasing the number of trials could help disambiguate the results more closely, but the number was dictated by the limitations imposed by high training times. In particular, the addition of the CRF layer caused a dramatic increase in the time needed to train the model (Table 4.8), presumably at least partially because its computation is performed on a CPU rather than a GPU, which is used by other parts of the model. Each training was executed with a patience of four epochs, i.e. it was stopped once the F-score on the validation set did not increase for four epochs.

In our experiments we followed the common practice of using GloVe embeddings. To assess their effect on the performance, we also trained the model using random initialization for the word embeddings (`bilstm-crf-cnn-random`). This resulted in a significant decrease in the score, confirming that semantic information encoded in word vectors is important for the success of the task.

We mentioned in §4.3.3 that our dataset is large compared with datasets for similar tasks. In order to align the results more closely with existing research, we investigated training the models with only 10% of the training data, randomly selected (`bilstm-crf-cnn-small`). The smaller version trained better with the stochastic gradient descent (SGD) rather than Adam used for the full model, and required an increase in patience to ten epochs. The reduction in the size of the training dataset resulted in the score dropping significantly to 0.812 (cf. Table 4.7), while the baseline model dropped to 0.674 (`bilstm-small`). We suspect that our framing of the surface semantic chunking problem is particularly data-intensive because of the length of the fragments to be learned and the associated sparsity of some labels.

A large proportion of functional fragments comprises closed-class words, such as coordinators. The challenge lies in distinguishing which occurrences of these words represent valid chunking opportunities and which fall within chunks. We had a look at the tokens most commonly participating in functional fragments and labelled as B-/I-FSUB or FCONJ. There is a high level of agreement between the tokens marked in this way in the test labels

Configuration	Training time
bilstm	47 min
bilstm-crf	48.5 hrs
bilstm-crf-cnn	42.5 hrs
bilstm-crf-cnn-bioes	46 hrs
bilstm-crf-cnn-small	5 hrs 40 min

Table 4.8 Training duration (for a single trial, including patience).

and those tagged by the model: 28 appear in both top thirty lists, with small differences in relative ordering. The precision and recall for inside labels (I) are lower than for start labels (B), e.g. 0.89 vs 0.63 for subordination. A likely cause is the fact that functional fragments have more than one token in the presence of modifiers, which tend to be sparse and varied in the training data. That effect compounds with the general relative sparsity of functional fragments compared with chunks due to the length of chunks in the training data. The F-score for other labels lies in the range of 0.82-0.96, with the best score for the inside label of coordination clauses.

4.4 Conclusions

As the experiments described in this chapter demonstrate, semantic chunking can be defined not only on complex semantic representations, but also on the surface form of sentences. The comparison with established related tasks indicated that a sequence labelling approach would be appropriate to model chunking directly without access to more complex representations of input sentences. The models from §4.3 demonstrate that state-of-the-art solutions to tasks such as NER and shallow chunking are indeed successful in finding surface semantic chunks. The results presented form a lower bound on the performance due to the lack of in-depth hyperparameter optimization.

The models were trained in a semi-supervised paradigm, with the training data created automatically based on the relationship between DMRS subgraphs and string fragments (§4.1). We monitored the quality of the dataset using an intrinsic evaluation procedure which simulates the behaviour of candidate surface semantic chunks in the target task of parsing (§4.2). The success of the labelling models is encouraging from the point of view of the flexible processing methodology that semantic chunking represents, as the automated chunking reduces the cost of the creation of dedicated chunking datasets, tailored to particular target tasks.

In Chapter 5 we will take steps to further reduce the human input required for creating chunks, as we propose a new chunking model which exploits inherent properties of representation to perform semantic chunking semi-automatically, given a set of pre-defined constraints. The resulting chunks represent a wider variety of grammatical structures and lead to more chunking opportunities compared to the rule-based system.

Chapter 5

Automatic detection of chunking opportunities

After investigating whether semantic chunking can be performed on the surface representation, we return to the topic of DMRS chunking. In Chapter 3 we described a prototype chunking system based on a set of hand-crafted rules, which identified semantic chunks corresponding to finite clauses. The chunker constituted the proof of concept for the usefulness of the task of semantic chunking and how it can be applied in a processing pipeline. Both the design approach and the form of chunks were highly restrictive, although sufficient for the purpose of preliminary experiments.

The investigation presented below expands the work from Chapter 3 with two primary goals in mind:

a) automated creation of chunking rules,

b) increased coverage and flexibility of semantic chunking.

Departure from hand-written rules is a necessary step for semantic chunking to achieve broader coverage and more flexibility. The rules of Chapter 3 cover a handful of manually selected language phenomena, defined on DMRS through inspection of sample parses. The structures used for chunking are strictly limited to those explicitly stated in the rules. Any expansion requires hard-coding new rules, which results in an unwieldy, brittle algorithm that is difficult to maintain as the number of covered phenomena and edge cases increases. A glimpse of this issue was exposed in the early experiments, where two major recurring sources of errors were constructions unaccounted for by the rules: sentential modifiers and the interaction between coordination and complement clauses (§§3.5, 3.6).

Furthermore, the rules are based on patterns of DMRS links and predicates corresponding to the selected structures in the particular version of the ERG. As new versions of the grammar

are released, the chunking rules become obsolete. Such a strong dependence on the grammar limits the applicability of the system to other grammars, languages and representations. The general principles of finding suitable trigger nodes and structures with valid chunks hold but the algorithm itself has to be largely rewritten.

The prototype approach was suitable as an illustration of the usefulness of semantic chunking as a pre-processing step, but it stands at odds with the broader design philosophy motivating the task in the first place. In Chapters 1 and 2 we introduced semantic chunking as a flexible way of reducing the computational cost of applications further down a processing pipeline. Semantic chunks should be adjustable to strengths and limitations of input representations and target tasks. Human annotation and rule definitions are too costly to serve as primary mechanisms behind the creation of dedicated chunking systems.

The main focus of this chapter is the introduction of a new chunking approach which takes steps to address the concerns outlined above. The new model relies on the internal structure of DMRS, in particular its representation of scopal dependencies. In §5.2 we summarise the principles of identifying semantic subgraphs suitable to act as semantic chunks. The automated extraction process allows us to limit the human input to defining initial constraints on the form of chunks and to filtering suggested chunkings based on the prior knowledge of the target task.

Before we can describe the chunking model, we need to have a more in-depth look at the properties of the scopal hierarchy in DMRS. In §5.1 we discuss in detail possible scopal configurations and how the hierarchy introduced in §2.3.2 can be expanded to include all types of scopal operators. In particular, we show how it can be organised in a tree structure, from which candidates for semantic chunks emerge naturally based on the branching of scopal links (§5.1). The resulting inherent structure becomes the foundation for our new chunking model (§5.2). Given a starting set of assumptions about the form of DMRS chunks, suitable subgraphs can be extracted automatically, even for previously unseen examples. Although the particular implementation is DMRS-specific, the algorithm itself can be adapted to other graph-based representations.

In order to generalize over the discovered chunking opportunities, we introduce **templates** (§5.3) – small, underspecified DMRS graphs which capture key structures participating in a given chunking decision. They can be used to investigate properties of chunks, as well as to filter chunking decisions based on the known target task requirements.

Semantic chunking is a pre-processing task which needs a target task for evaluation. In §5.4 we return to realization experiments to see how the loosening of constraints on the form of chunks affects the task performance. We also propose another way of incorporating the

information stored in functional subgraphs into the assembled surface of the full sentence, leveraging the generalized information stored in templates (§5.4.4).

5.1 Scopal hierarchy

In §2.3 we described the details of DMRS representation, in particular when used with the ERG. Nodes in a DMRS graph are connected by two major types of links, scopal and non-scopal (§2.3.2), which reflect the nature of the connection between a node and its arguments. Nodes with scopal links (other than quantifiers) are of special interest to chunking because they behave as if acting on entire semantic subgraphs rather than individual nodes (§2.3.4). They are always event nodes (§2.3.1), and so are their scopal arguments.

In §2.3.2 we touched upon the concept of a hierarchy of scopal operators within a DMRS. Borrowing from Rhetorical Structure Theory (RST; Mann and Thompson (1988), §2.1.4), we can think of semantic chunks in a sentence in terms of a nucleus and its satellites. The central node of a DMRS (§2.3.3) belongs to its nucleus chunk, which determines its overall grammatical properties. In particular, a DMRS representation of a sentence has a finite node at its centre (§3.1). RST makes a special case for sequence and contrast relations, in which the connected constituents are on equal footing and are both classified as nuclei (Mann and Thompson, 1988). Our adapted use of the terminology requires an unambiguous ordering, and we make a general assumption that the left coordinate is the nucleus of the coordination. The decision is supported by the fact that in VP coordination this is the coordinate with uninterrupted syntactic connection to the shared subject and in the ERG 1214 the coordinator shares the grammatical properties of its L-INDEX argument.

So far, we discussed the scopal hierarchy based on the bracketing system of a traditional logical representation, which only determines the ordering of unary operators (§2.3.2). In this chapter we demonstrate how the hierarchy can be expanded to take the form of a tree, with meaningful left and right branchings. We consider possible configurations of scopal trees based on these assumptions, and discuss how they relate to the concept of situations and other theoretical foundations of semantic chunking outlined in Chapter 2, with the focus on finding nodes which signal a chunking opportunity.

The tree representation originates from an observation that non-unary operators, in particular subordinating conjunctions, always take two scopal arguments with fixed roles: ARG1/H for a main clause and ARG2/H for a subordinated clause, while unary operators, such as adverb-like phrases, always act on their ARG1/H argument. We can form a scopal tree by assigning dominant ARG1 arguments to left subtrees and subordinate ARG2 arguments to right subtrees (L-HNDL and R-HDNL for coordination). As a result, we discover that

the treeified scopal hierarchy exposes central nodes of semantic subgraphs (§2.3.3) with event-type variables (§2.3.1). We previously discussed such subgraphs as the best candidates for semantic chunks (§§2.3.1,3.1) but did not have a reliable way of distinguishing them from other event nodes. In this chapter we refer to central nodes of chunk subgraphs as **chunk centres** to distinguish them from other central nodes.

Let us consider an example based on a sentence from [1]:

(85) BLEU is designed to approximate human judgement and does not perform well if used to evaluate isolated sentences.

The centre of the DMRS (Fig. 5.2) is the node `_design_v_1` in the nucleus clause *BLEU is designed*. The diagram in Figure 5.1 illustrates the scopal relationships as a tree. Dominant ARG1 arguments corresponding to main clauses and subjects of modality operations fall into left subtrees, while satellite clauses branch out to the right. Rectangular nodes of the tree correspond to individual situations:

_design_v_1 BLEU is designed.

_approximate_v_1 BLEU approximates human judgement.

_perform_v_1 BLEU does not perform well.

_use_v_1 BLEU is used.

_evaluate_v_1 Individual sentences are evaluated.

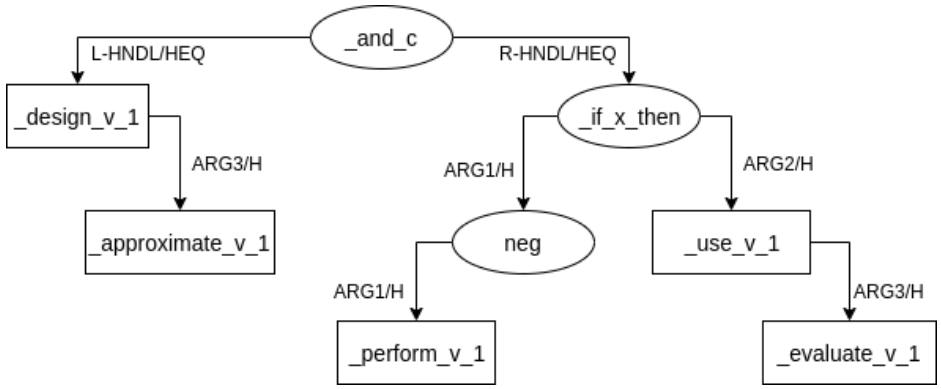


Fig. 5.1 Scopal tree for Example 85.

[1] Ident 10090130

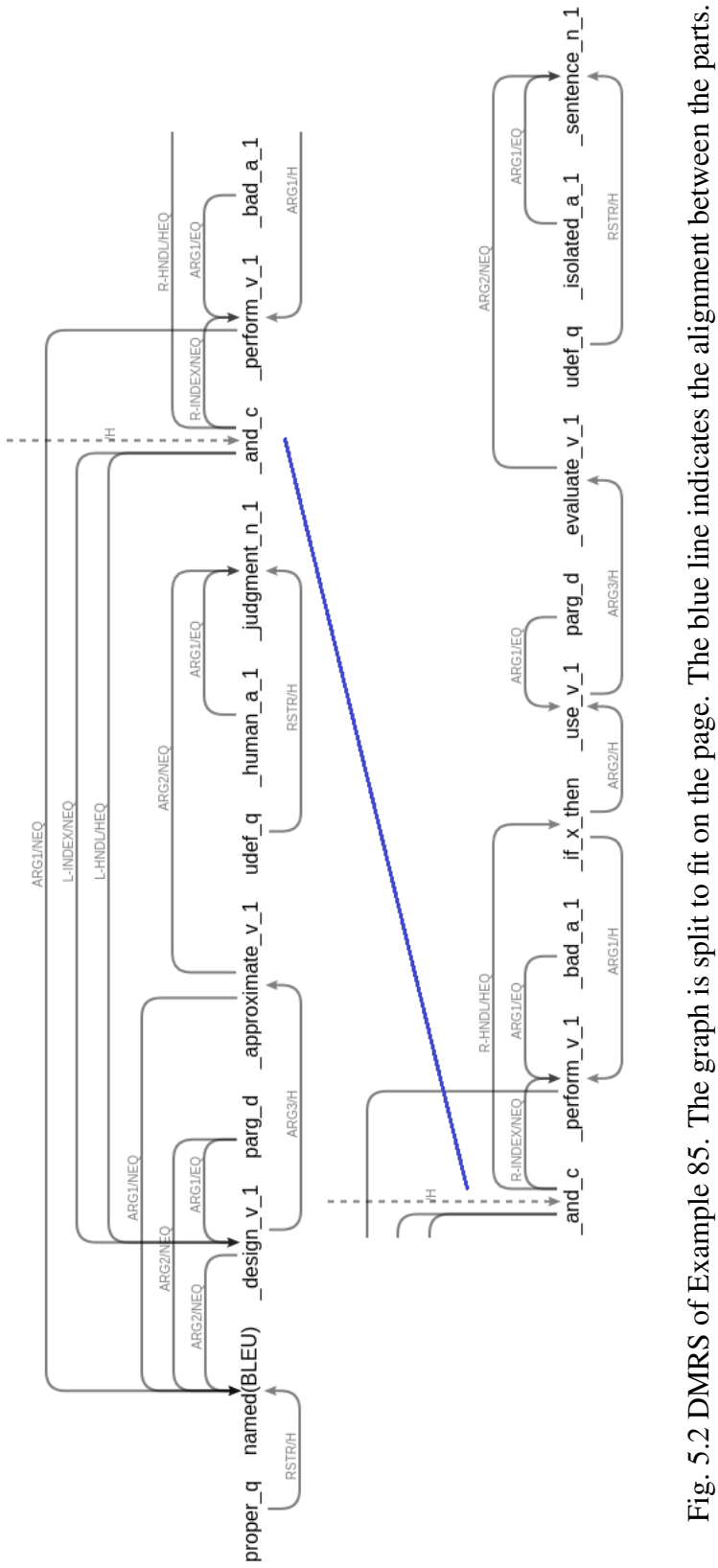


Fig. 5.2 DMRS of Example 85. The graph is split to fit on the page. The blue line indicates the alignment between the parts.

5.1.1 Configuration 1: Single scopal ARG1.

There exists a large category of nodes with a single scopal argument in the ARG1 slot. These nodes can be grammar predicates or real predicates (§2.3.1) corresponding to adverbs or verbs, and they act as operators with limited semantic content independent of their argument. Some examples are presented in Figures 5.3. They are often modal verbs, like *must* in the example in Figure 5.3c, but not exclusively. In a simplified conventional logical notation we would express such relations as:

(86) The mission will probably be a disaster. ↔ `probably(be(mission, disaster))`

(87) Frodo must destroy the Ring. ↔ `must(destroy(Frodo, the Ring))`

Even though nodes with a single scopal ARG1 can be tense carriers and top index nodes of subgraphs, they have insufficient autonomy to act as chunk centres and always belong to the same chunk as their arguments. This is in line with the treatment of modals and negation in situation semantics (Barwise and Perry, 1983), where they are considered constraints on situations they modify. From a practical standpoint, in *MRS they have no arguments other than the verb they modify, which means semantic chunks associated with them would comprise a single node and as such, do not meet the granularity requirements of chunking (§2.1.3).

Let us consider the following sentence:

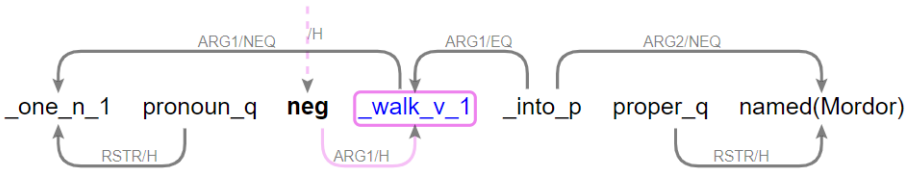
(88) Isildur failed to destroy the Ring.

Its DMRS graph is shown in Figure 5.3d. The verb *failed* illustrates the scenario in which one-to-one correspondence between verbs and situations (§2.1.2) is violated, and is a good example of how the form of semantic chunks must account for the input representation. Based on the surface string of the sentence alone, we could interpret the example as expressing two situations:

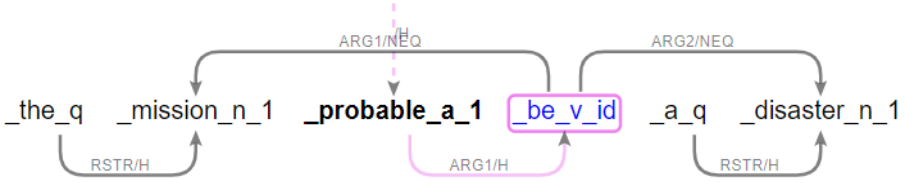
a) a "real" situation with Isildur present and the Ring remaining intact, where the action/event of failing occurred,

b) a situation in a hypothetical world where Isildur destroyed the Ring.

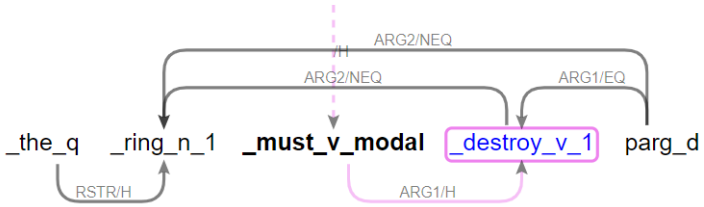
The first situation is not fully supported in the DMRS representation, as the `_fail_v_1` predicate takes only the eventuality of destroying as its argument, acting as a modal operator. The entities of Isildur and the Ring are only arguments of the latter situation. The verb *fail* belongs to a special class of subject-raising verbs, which have their own lexical type category in the grammar lexicon (§2.3.1). The design of their treatment in the ERG supports our approach.



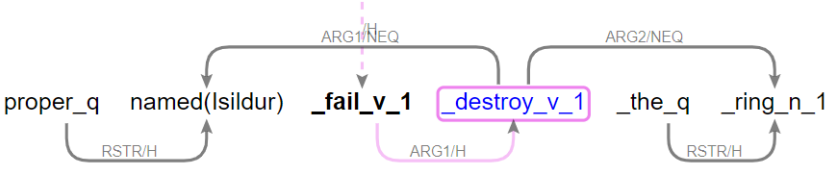
(a) *One does not walk into Mordor.*



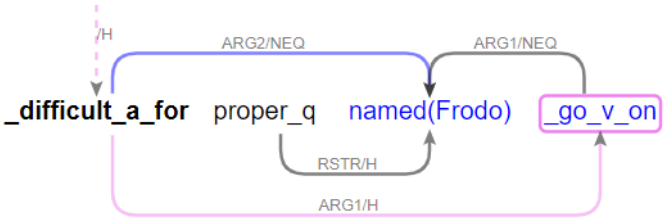
(b) *The mission will probably be a disaster.*



(c) *The Ring must be destroyed.*



(d) *Isildur failed to destroy the Ring.*



(e) *It is difficult for Frodo to go on.*

Fig. 5.3 Examples of DMRS with scopal operators.

An interesting form of Configuration 1 arises when a node with a scopal ARG1 argument has another non-scopal argument. An example of such a DMRS is shown in Figure 5.3e:

(89) It is difficult for Frodo to go on.

The construction under consideration is a type of control relation (cf. §5.1.3). The comparison of the ERG1214 analysis for this example with others sharing the ARG1/H configuration suggests that the phrase *it is difficult for Frodo* should be interpreted as a modality operator for the core situation of Frodo going on, no different that a modal construction *it is possible that*. The sentence can be paraphrased as Example 90, although the paraphrase removes the explicit control relation between Frodo and going on:

(90) Going on is difficult for Frodo.

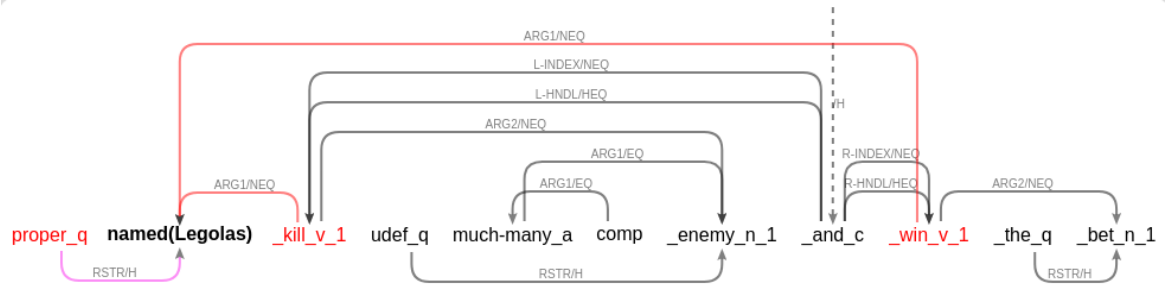
The gerund *going on* becomes the subject of the sentence, more clearly than in the original phrasing with the expletive *it*. If a phrase expresses a situation with another situation as its vital participant/argument, can it ever be sufficiently separated to satisfy the semantic chunk constraints? The ERG analysis which parallels that for modality operators suggests a negative answer, at least for the ERG representation.

5.1.2 Configuration 2: Two scopal arguments.

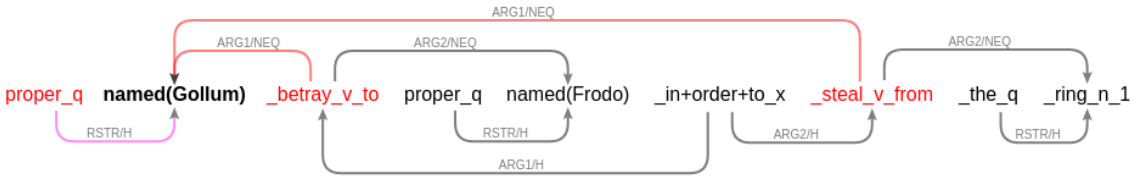
We already encountered a subset of nodes with two scopal arguments as trigger nodes for coordination and subordinating constructions in the prototype system of Chapter 3, although only in the limited scenarios yielding finite clauses. The root of a subtree with this configuration is an operator relating two situations, and therefore two chunk candidates, but it itself does not belong to either. In order to preserve information about how the chunks are connected, we have to preserve the root operator and its links in a separate functional subgraph (cf. §3.3).

The two scopal arguments sometimes share their arguments (examples in Fig. 5.4). Unlike the operator, the shared argument belongs semantically to both semantic subgraphs at once. At first glance, this violates the principle of self-containment we imposed on semantic chunks from the outset, but we can preserve the information about the connection between chunks and their arguments. One way would be to duplicate the semantic subgraphs of the shared arguments, so that each chunk maintains its own copy. The approach is impractical in the face of potentially large argument subgraphs and runs against the goal of improved processing efficiency.

Another way, which we implement in our system, is to assign the argument to one of the chunks and store the information about the severed connection in the functional graph



(a) *Legolas killed more enemies and won the bet.*



(b) *Gollum betrayed Frodo to steal the Ring.*

Fig. 5.4 Examples of DMRS graphs with nodes in configuration 2 sharing arguments.

created for the purpose of preserving the operator. The final form of the functional subgraph associated with each chunking decision contains all links between a nucleus chunk and its satellite chunks, together with copies of nodes directly participating in inter-chunk links. According to our hypothesis about the nature of scopal links, the nucleus chunk of an utterance always lies in the left scopal subtree (§5.1). As such, the chunk leftmost in the scopal order tree of the subgraph under consideration retains custody of any shared arguments. For VP coordination, it is the left coordinate, while in control constructions it is the main clause.

5.1.3 Configuration 3: Non-ARG1 scopal arguments.

Configuration 3 is the only configuration which allows the root of the scopal tree to be a valid chunk centre. We encountered it previously during prototype chunking, because some verbs and adjectives use an ARG2 or ARG3 scopal argument to represent a clausal complement (§3.2.3). Just as for Configuration 2, the restrictions on the form of chunks from Chapter 3 no longer hold in our current consideration.

Apart from finite complements discussed previously, non-ARG1 scopal arguments play a role in control constructions, a type of subordination in which the main and subordinate clause share an argument. An example of a DMRS graph with a control construction is shown in Figure 5.5). It corresponds to the sentence:

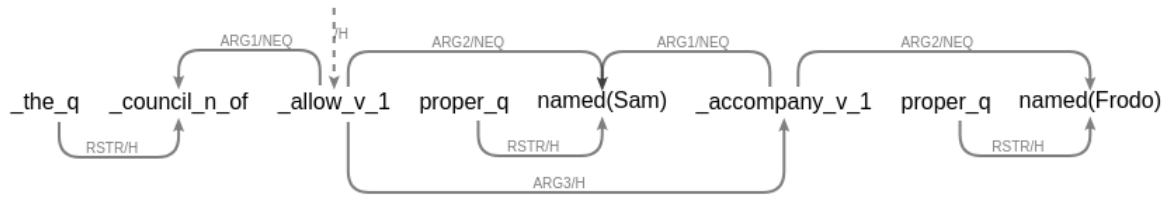


Fig. 5.5 DMRS for *The Council allowed Sam to accompany Frodo.*

(91) The Council allowed Sam to accompany Frodo.

The relationship between the operator and the central node in the example from Figure 5.3e is also a case of control:

(92) It is difficult for Frodo to go on.

In that scenario we argued earlier that the subgraph equivalent of *it is difficult for Frodo* is not a separate chunk, because it serves to modify the going-on situation. In Example 91, however, there are distinctly two situations:

1. a situation with the Council and Sam as participants at the time when the permission is given,

2. an outcome situation with Sam and Frodo as participants.

The problem of shared nodes can be addressed in the same way as for Configuration 2, with main clauses retaining the argument.

Generally, we do not treat adjectives and prepositional phrases as introducing individual situations sufficient for semantic chunking (§2.1.3). The ERG analysis of sentences like Example 93 (Fig. 5.6) indicates that an exception should be made for sentences such as:

(93) Bilbo put the Ring in his pocket.

The preposition is not tensed, but introduces its own outcome situation, just as a verb would (*accompany* in Example 91). The situation in which Bilbo's pocket contains the Ring is distinct in time, location and participants to the situation with Bilbo moving the Ring. Accordingly, the ERS treats *in his pocket* as a scopal control argument of the verb *put*.

Another special scenario is associated with coordination. Since a coordinator node can be both the top and top index of a DMRS, examples with only a right coordinate still receive a full sentence analysis, e.g.

(94) But Sam could carry Frodo.

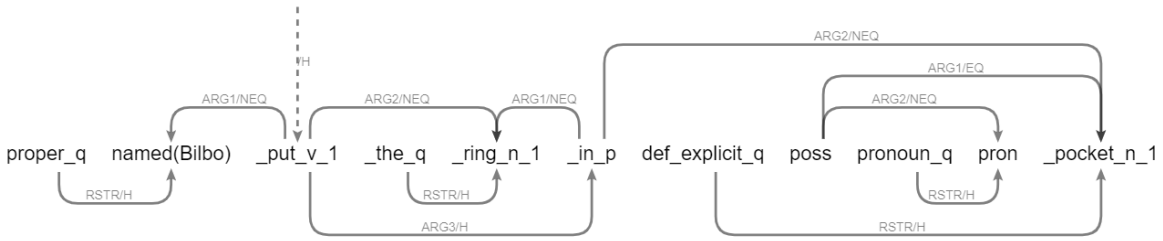


Fig. 5.6 *Bilbo put the Ring in his pocket.*

As far as we can tell, this is the only scenario where the top of a DMRS is not a chunk centre and has no left scopal subtree. The missing left operand shifts the pattern from Configuration 2, where the coordinator node would be assigned to the functional graph of the nucleus chunk. Due to a lack of other options, the right coordinate in this scenario retains the operator.

5.2 Chunking based on scopes

The survey of scopal relationships in DMRS graphs provides us with a basis to formulate general principles of how situations are represented in DMRS. The result of our observations is a new scope-based chunking system which we introduce in this section.

Semantic subgraphs based on individual situations are prime candidates for semantic chunks, provided we can identify them reliably. The criterion we relied on so far is the tense property, i.e. any tensed node is a valid chunk centre. The branchings of the scopal hierarchy help us identify **chunk centres**, i.e. eventuality nodes that are central nodes of semantic subgraphs representing situations, regardless of their grammatical properties. In general, we expect chunk centres to occupy leaves of the scopal tree. We make the following assumptions regarding valid chunk centres:

a) Nodes with left subtrees are not chunk centres. Instead, they modify their left child tree (Config. 1, §5.1.1).

b) Leaves of the scopal tree are chunk centres.

c) Nodes with only right subtrees are chunk centres if their predicates have a verb, preposition or the adjective/adverb ERG part-of-speech tag (Config. 3, §5.1.3).

Chunks in the prototype system of Chapter 3 interacted only via trigger links and each chunk was connected to the trigger via a single gatekeeper node. We introduced this constraint to ensure that the chunks are self-contained. In this chapter we investigate what happens

when no such limitations are imposed on the chunking process and whether information can be preserved for a wider range of DMRS subgraphs.

Let us consider a DMRS graph with a scopal ordering represented by a tree T. A chunking opportunity appears whenever T branches out to the right. If a scopal node has no right subtree, it is only an internal modifier of the nucleus chunk. Let T_n be the subtree of T with a root at node n. If n has a right subtree, one of the following three conditions applies:

a) n is the centre of the nucleus of the subgraph covered by T_n (Config. 3, §5.1.3)),

b) n has a left subtree and its leftmost node is the nucleus centre (Config. 2, §5.1.2),

c) n is a coordinator with only a right coordinate (special case of Config. 3, §5.1.3).

The chunking opportunity between the root and its right subtree has to be ignored in the special case of Configuration 3, since there is no other chunk which could include the coordinator root node.

The main two scenarios require a new satellite chunk to be created. The central node of the satellite is the leftmost valid chunk centre in the right subtree of n. Let us call the satellite centre s (Fig. 5.7). The satellite chunk covers all the nodes in the scopal subtree with the root r, and includes any nodes disconnected from n by the removal of r. As a result, the leftmost chunk in the scopal tree retains any arguments shared with other chunks. The special case of Configuration 3 occurring for a partial coordination means that the top of the new chunk, r, is the rightmost parent of s, but not necessarily the immediate right child of n.

The nodes of the newly formed chunk are removed from the nucleus subgraph, but only after the connection between the two chunks is recorded in a functional graph. The functional graph contains all links (a,b) such that a is in the nucleus chunk and b in the satellite. It also stores copies of nodes needed to represent the links. The new chunk formed this way has a potential to be chunked further, so the chunking proceeds recursively until we reach the leaves of the scopal tree.

Chunking in the presence of the b) scenario requires an additional step. Node n, i.e. the root of the subtree under consideration, does not belong to either chunk. Instead, it is stored in the functional graph, together with any nodes disconnected from the nucleus chunk if n is removed. The root of the left subtree of n becomes the new top of the nucleus and another target of recursion.

5.3 Templates

Thanks to the chunking algorithm described in the previous section, we can divide any DMRS into semantic chunks, even based on structures we have never observed before. The

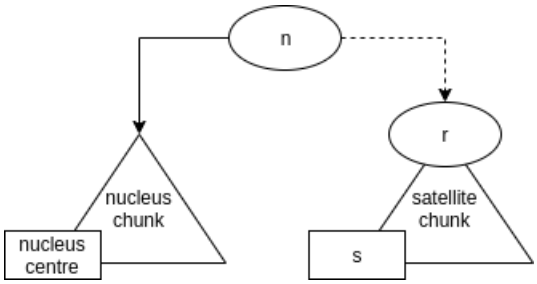


Fig. 5.7 Schematic structure of the scopal tree in Configuration 2. Dashed line indicates that *r* is not necessarily the immediate child of *n*. In Configuration 3, *n* is the centre of the nucleus chunk instead.

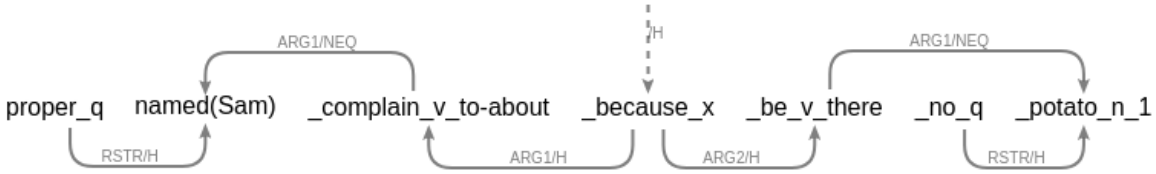
resulting chunks represent a wider variety of grammatical constructions and take more varied forms compared with the chunks from Chapter 3. This flexibility fits well with the task characteristics, but in order to inspect the chunks and to impose constraints specific to the intended application of chunks, we need a way of generalizing across our examples. The chunking system from Chapter 3 solved the issue by starting with a set of rules, strongly limiting what chunks can be formed. For the new model we propose a different approach that better suits the adaptable nature of chunking.

The prescriptive rules of the prototype system are replaced in the new model by descriptive **templates** extracted from a training set. A template is a small underspecified DMRS based on a functional graph associated with a chunking decision. It captures relationships between interacting chunks. Individual rules from §3.2 find their equivalents among the automatically extracted templates (§5.4.3), but while the original rules were hard-coded generalisations, templates emerge naturally from the basic principles of the representation and the grammar. We demonstrate the benefits of the new paradigm in §5.4.4, where the automatically extracted patterns generalizing across the dataset help us address the issue of how to represent functional subgraphs in realization with chunking (§5.4).

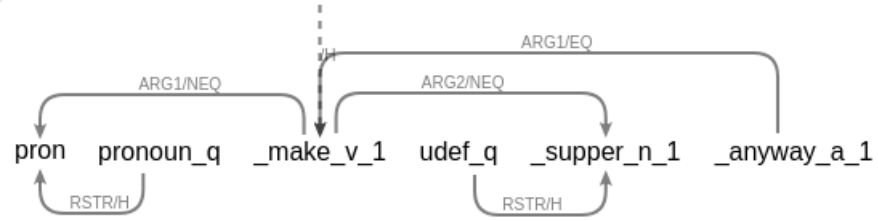
Templates are directly linked to functional graphs. Each functional graph stores nodes and links which do not belong to either the nucleus or satellite semantic chunks. In order to store the links, it also includes copies of nodes which participate in the connection from within the chunks. These nodes provide constraints on what type of DMRS subgraphs can participate in the particular chunking pattern. For example, the functional graph and two chunk subgraphs from Figure 5.8 are the result of chunking Example 95 based on the clausal coordination:

(95) Sam complained because there were no potatoes, but he made supper anyway.

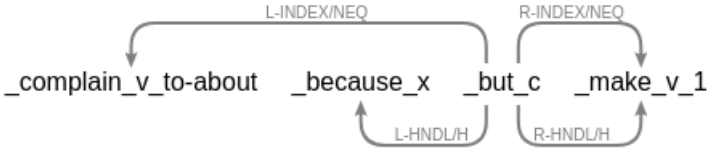
The functional graph would take a different form if the left coordinate was a simple clause. At the same time, chunking other sentences with a similar coordination would yield functional



(a) The left coordinate chunk of Example 95: *Sam complained because there were no potatoes.*



(b) The right coordinate chunk of Example 95: *he made supper anyway.*



(c) The functional graph for Example 95.

Fig. 5.8 The chunks and the functional graph for Example 95

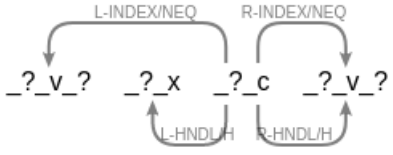


Fig. 5.9 A template based on the functional graph from Fig. 5.8c.

graphs that are almost identical, with the exception of properties of nodes, such as their lemmas or tenses.

This observation allows us to generalise across chunking decisions and find reliable patterns. Each functional graph can be converted into a delexicalised template, shared between chunking points based on similar DMRS structures. To create a template, we convert each node in a functional graph into a delexicalised **nonterminal**. A potential template for Example 95 is shown in Figure 5.9.

The type of nonterminals can be adjusted to match the level of generalization allowed by the target task and input representation. Our application of choice is realization from DMRS, and in §5.4.2 we give an example of how the nonterminals can be tailored to the needs of that particular task and how they reflect the constraints imposed on the form of semantic chunks (§5.2).

In §5.4.3 we present the templates extracted from our training set. The scope-based chunker found 2056 chunking opportunities, which can be represented by 789 templates, some of which have an almost one-to-one correspondence to the rules from the prototype model. The generalising power of the template approach is demonstrated by the fact that 303 of the extracted templates account for over 70% of the chunking decisions encountered in the test set.

5.3.1 Related work: HSST/R

Expressing generalizations across chunking decisions in the form of templates was inspired by the synchronous context-free grammars (SCFG) used in hierarchical phrase-based translation systems (§2.2.6), in particular, Hierarchical Semantic Statistical Translation (HSST) and Realization (HSSR) (Horvat, 2017; Horvat et al., 2015). HSST is a hybrid translation system, combining Hiero (Chiang, 2005, 2007) (§2.2.6) with information from semantic parses. It adapts the SCFG to comprise graph-to-string rules, using the same semantic representation and grammar as the work in this thesis – Dependency Minimal Recursion Semantics (DMRS; §2.3). HSSR (Horvat et al., 2015) is an adaptation of HSST to statistical realization.

The HSST grammar rules connect DMRS subgraphs of a sentence in the source language to surface fragments of the target sentence, rather than linking two surface sentences directly. The initial set of source rules is extracted from suitable subgraphs of the input graph. Each rule is then iteratively subtracted from other rules wherever possible to create non-terminal rules. The process is similar to the one we use for chunking, where semantic chunks lower in the hierarchy are removed from their source graphs and processed recursively. Each non-terminal in a template stands in for either a simple chunk or another complex hierarchy.

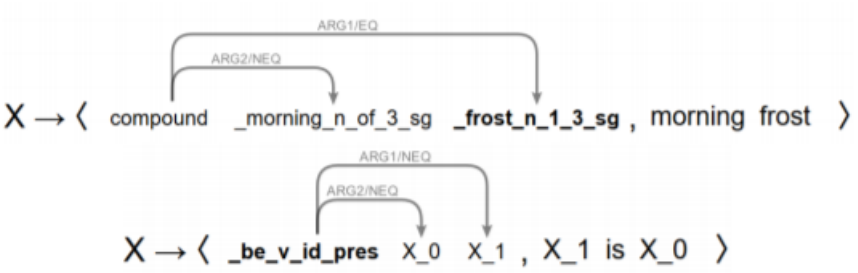


Fig. 5.10 An example of a terminal (top) and non-terminal (bottom) realization rule for HSSR (Horvat, 2017).

Non-terminals in the rule subgraphs have matching non-terminals in the surface side of the rule. Figure 5.10 shows examples of rules with and without nonterminal symbols.

The graph sources of the SCFG rules are a variant of semantic subgraphs. Horvat (2017) originated the term but defined it through the extraction algorithm. The resulting subgraphs are the precursors of the entities we discuss more conceptually in Section 2.3.4. The HSST/R rules are based on DMRS graphs simplified to polytrees, i.e. directed graphs which resemble trees but have multiple root nodes. Instead, we extend our use of the term "semantic graphs" to general graphs, changing the details of the definition but preserving the intuition behind it.

5.4 Realization with templates

Semantic chunking is a practical task defined on a per-target-application basis. Like the prototype system, the new chunking model should not be evaluated in a vacuum. In this section we apply the wider variety of automatically extracted chunks to the previously visited task of realization from DMRS.

Realization with chunking comprises several steps, similar to those used in §3.6:

a) realize individual chunks,

b) create surface representations of the functional graphs,

c) assemble the full surface of the sentence.

We discuss how to realize the newly extended variety of chunks in §5.4.1. As before, the main issue in the chunk-based processing is the treatment of elements in functional graphs associated with each chunking decision. Limited connections between chunks produced by the prototype system, together with their simple form, allowed us to simulate chunks with simple placeholders (§3.6.1). We found, however, that the approach does not scale well

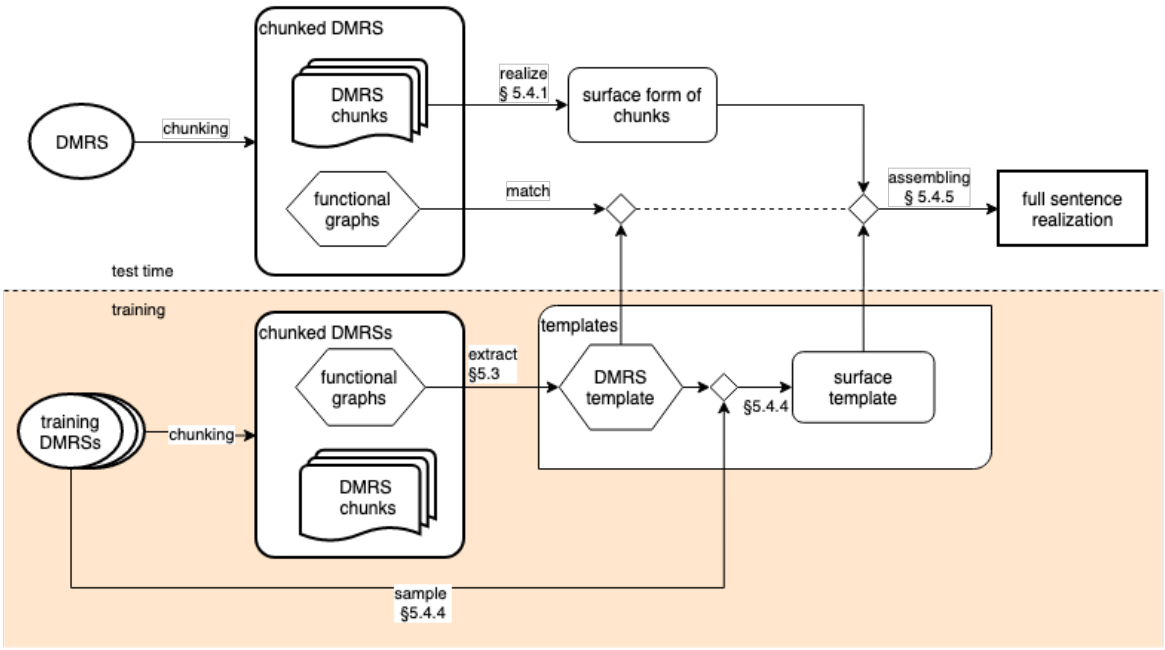


Fig. 5.11 Realization with chunking using surface templates.

with the increased variety of chunk types. As an alternative, we propose to convert DMRS templates based on functional graphs from a training set into **surface templates** (§5.4.4), which can be applied to matching chunking opportunities in new examples. Together with the chunking constraints described in §5.2, the creation of surface templates guided the design of nonterminals (§5.4.2). We present the resulting templates extracted from the dataset in §5.4.3, followed by the results of their application to realization with chunking in §5.5. In the final step we assemble the surface templates and chunk surface forms into the full string representation of the input sentence (§5.4.5).

Figure 5.11 summarises the updated process of realization with chunking. The training time (bottom part of the figure) is dedicated to the creation of surface templates from the training data. At test time (top part) we attempt to match each functional graph with an existing template produced during the training. If successful, the match identifies the surface template that should be used to represent the functional graph in the final surface string. The final step (rightmost in the figure) assembles the surface templates and realization results for individual chunks (top path of the top part of Fig. 5.11) to produce the full realization of the sentence.

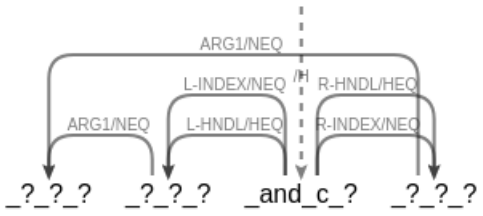


Fig. 5.12 Template for the VP *and* coordination.

5.4.1 Realizing chunks

Let us first consider how to realize an individual chunk. The ERG can generate surface forms from *MRS representations of full sentences, or suitably framed fragments (§2.3.5). The experiments in §3.6 were limited to the former type of chunks, but here we aim to accommodate a wider variety, which requires some chunks to be modified before they can be realized. Modifications needed depend on the type of chunk as determined by grammatical properties of its central node.

Modality operators from Configuration 1 (§5.1.1) break this mould and directly impact the grammatical functionality of the chunk. The central node of the phrase *I can swim* is the untensed verb *swim*; the tense property is determined by the modal *can*. To account for this, we treat the central nodes as if they inherit node properties from their scopal predecessors in Configuration 1. As a result, *I can swim* can be correctly recognised as a finite clause. Only the predecessors assigned to the chunk are taken into account.

Finite chunks can be generated in their original form, unless they miss an argument shared with another chunk, e.g. the right coordinate in the template from Figure 5.12. Such shared arguments are always instance nodes (§2.3.1), and for the purpose of realization they can be replaced with a predetermined placeholder of appropriate plurality. After successful generation, we can subtract the placeholder string from the final result to yield the chunk surface (cf. placeholders in §3.6).

In this set of realization experiments we use an ERG 1214 grammar file enhanced by additional trigger rules[2] (§2.4.1). Even with the new additions, we found that the ERG1214 does not generate reliably from infinitival verb phrases[3]. As a bypassing measure, we use the property of English according to which the modal verb *can* takes as its argument an infinitival clause. In order to reliably generate from an untensed verb phrase, we enhance it with a present tense top node `_can_v_modal` and, if needed, placeholders for any shared

[2]Provided by Dan Flickinger.

[3]For example, *predict an outcome* fails, but *predict an outcome with certainty* succeeds.

arguments encoded in the template. A post-processing step removes the surface strings of any added elements.

Chunks with progressive or perfective centres require different adjustments. Gerund chunks tend to generate as *to be doing something* over *doing something*. Similarly, passive untensed chunks without a passive subject can sometimes generate as *to be done*, while *done* is preferred in most of their occurrences. In both cases, we remove the extra string *to be*. When needed, it is accounted for by surface templates. It is possible for a gerund phrase to have a subject (ARG1/NEQ argument) outside of control scenarios, e.g. Figure 5.13. Realizing such a chunk requires converting it into a nominalized phrase, with a fragment top node (§2.3.5).The fragment modifier is also needed to realize untensed chunks corresponding to prepositional and adjectival phrases (Fig. 5.15). Whenever an adjective or preposition node has a perfective or progressive flag, its chunk can be processed like an equivalent verb chunk.

5.4.2 Nonterminals for realization

Nonterminals are underspecified nodes used in templates to represent entire semantic subgraphs. They can correspond directly to a particular node in the source DMRS, but in general they summarise overall properties which can be distributed across the subgraph.

Properties of chunks are mostly determined by their central nodes (§2.3.3), but the chunk centre itself does not have to participate in the functional graph if it has a scopal operator. When present, the operator is the top node of the chunk subgraph and serves as the hook in the *MRS composition system (§2.3.2), but unlike the central node, it has a limited impact on the chunk properties. For instance, an extra adverb in a subordinate clause should not affect the form of a template in most scenarios.

Whenever one of the chunk centres does not participate directly in a functional graph, we identify its closest explicitly present scopal parent. The corresponding nonterminal is then based on the central node instead. The fact that the top node of a subgraph is its point of contact with scopal operators is a fixed property of DMRS, guaranteeing the correct reconstruction of the full representation based on the generalised template.

To illustrate the issue, let us compare the following examples:

(96) Sam complained because there were no potatoes, but he made supper anyway (Example 95).

(97) Gollum cried and raged because Sam cooked the fish. (Fig. 5.14).

Both of these examples represent the interaction of a coordination and a subordinating conjunction. For the first one, both the chunk centre of the left coordinate and its top node

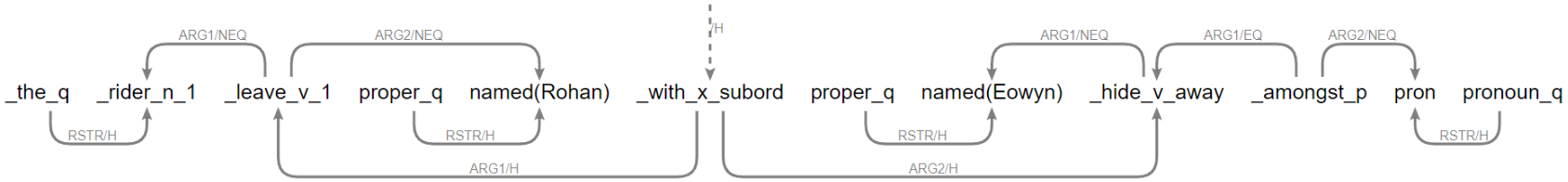


Fig. 5.13 DMRS for *The riders left Rohan, with Eowyn hiding amongst them.*

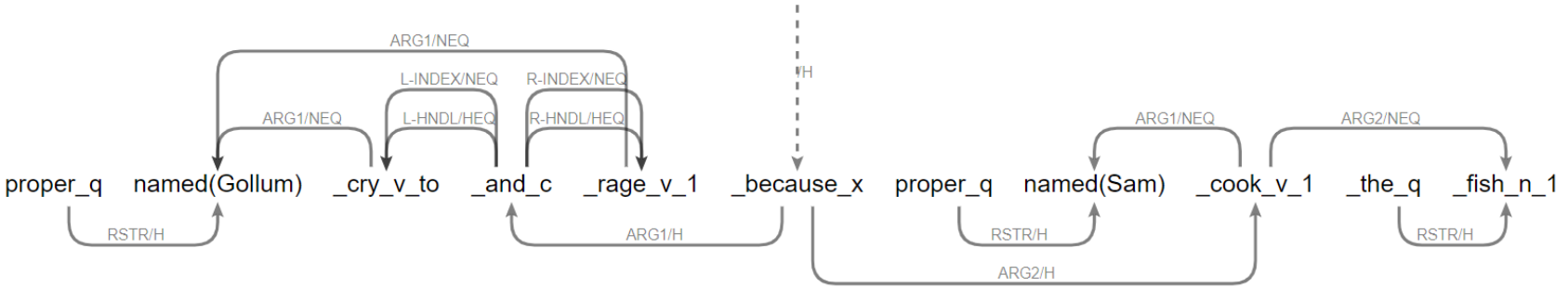


Fig. 5.14 DMRS of *Gollum cried and raged because Sam cooked the fish.*

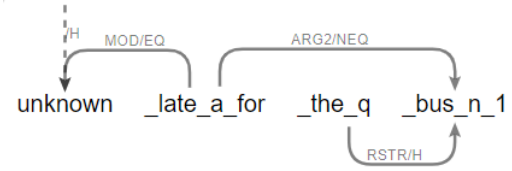


Fig. 5.15 DMRS for a fragment *late for the bus.*

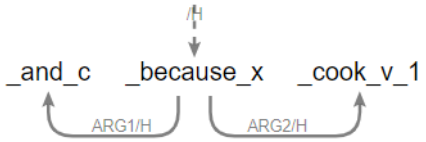


Fig. 5.16 Functional graph for Example 97.

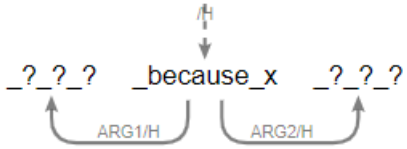


Fig. 5.17 Template for subordinating conjunction with *because*.

participate in the functional graph and the associated template (Fig. 5.9) because of the double coordination links. On the other hand, the functional graph of the second example uses only a single node from the main clause (Fig. 5.16). If we directly delexicalise the functional graph, we get a template in which the main clause chunk is represented by a coordinator node. This is, however, a missed generalization opportunity. Chunking Example 98 requires the same operations but would be represented by a different template:

(98) Gollum cried because Sam cooked the fish.

The left coordinates in both examples are finite clauses as determined by their central nodes (`_cry_v_to`). We can describe them by a single template (Fig. 5.17), which includes a nonterminal based on a delexicalised chunk centre, instead of the literal node participating in the represented link.

Although it is possible to work with fully delexicalised templates, our approach to realization requires a partially lexicalized version, because inter-chunk operators are only realized through surface templates based on training data (§5.4.4). Even though the two templates in Figures 5.17 and 5.18 could be merged by replacing their top operators with a nonterminal, e.g. `_?_x_?`, their corresponding surface templates have to be differentiated: *A because B*/*Because B, A* and *A if B*/*If B A*. We also retain the original predicate information for chunk centres which introduce other chunks as their scopal operators (Config. 3, §5.1.3). This allows us to distinguish nodes with compulsory scopal arguments, which we previously recognised based on their entries in the grammar lexicon (§3.2.3).

Within these constraints, we attempt to maximize the level of generalization. All tensed preposition, adverb/adjective and verb nodes are fully delexicalised, apart from an underspec-

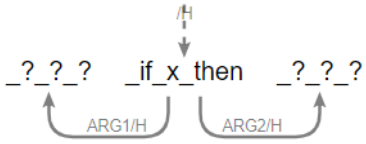


Fig. 5.18 Template for subordinated clauses introduced by *if*.

ified `tensed` value as their tense property, because they are centres of standard finite clauses that are grammatically equivalent. Other tensed nodes also preserve their part-of-speech tag if they have real predicates, or their predicate name for grammar predicates. Untensed nodes retain all of their node properties during the nonterminal conversion, with the exception of underspecified lemma and sense for real predicates. We also fully delexicalise all nodes with instance variables (§2.3.1). Since they do not contribute to chunking decisions, they are stripped of all specific predicate and node properties, other than the type of intrinsic variable.

We accommodate the impact of modality operators on the grammatical properties of chunks by allowing nonterminals of central nodes to inherit node properties from their in-chunk ARG1 scopal predecessors (cf. §5.4.1).

Modifiers of inter-chunk operators are a recurring source of complication in semantic chunking (cf. §3.5). They can be of unrestricted forms and sizes, inflating the number of potential templates without contributing much information about the chunked construction. We limit their impact on templates by removing all but the topmost node of their subgraphs from the template. For example, if a coordination node is modified by a prepositional phrase, the corresponding template will only retain a delexicalised preposition node. We base this simplification on the assumption that similar modifiers interact with their arguments in the same way (cf. §5.4.4). Degree modifiers applied directly to operators, e.g. in *only because*, are not delexicalised and are an integral part of a template because of their contribution to surface templates (§5.4.4). They can be readily recognised because of the shape of their predicates (`_?_x_deg`).

Construction	Count
_and_c (tensed)	114
_but_c (tensed)	40
implicit_conj (tensed)	36
_in+order+to_x	33
_and_c (untensed)	28
subord	27
_as_x	23
_when_x	20
_or_c	17
implicit_conj (untensed)	13

Table 5.1 Operators with the largest variety of templates

Construction	Count
subord	78
clausal _and_c	65
clausal implicit_conj	52
_in+order+to_x	48
_while_x with a finite clause	34
be with a clausal complement	32
clausal _but_c	32
VP _and_c	31
_if_x	30
_although_x with a finite clause	30

Table 5.2 Templates with the most applications

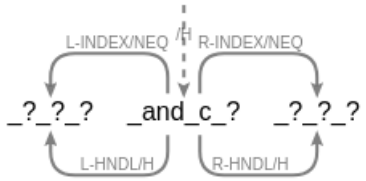
5.4.3 Extracted templates

The training set comprises a randomly selected 50% of the WeScience dataset (§3.4), with the other 50% reserved for testing. The domain of our experiments is limited to DMRS representations of full sentences, which eliminates 114 examples in the training dataset assigned a fragment analysis, indicated by the unknown node. 60 more examples are untensed utterances and further 61 graphs did not meet our scope assumptions. 1323 DMRSs out of the remaining 3229 DMRS graphs were chunked 2056 times, in patterns described by 789 templates.

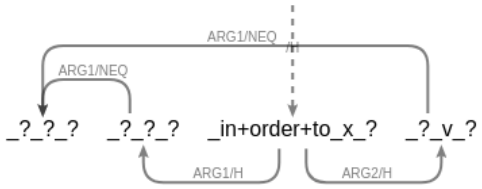
We organize the templates by the type of the main operator used for chunking. In the scopal tree order, it is the lowest tree node shared by the nucleus and its satellite. Table 5.1 lists the operators associated with the largest number of templates. Most of the 262 main operators correspond to a unique template. For example, each verb with a clausal complement has its own template under the delexicalisation rules we adopted. Although generalizing across multiple operators is possible, it does not work with surface templates (§5.4.4).

Table 5.2 lists the ten most commonly applied templates. They are all among the simplest possible templates, aligning with the intuition that the more elaborate the pattern, the less common it is. Visualizations of some of them are shown in Figure 5.19.

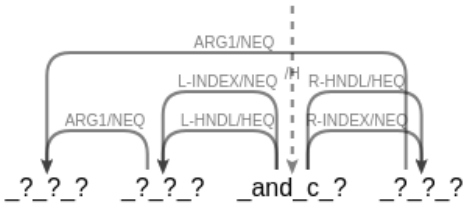
Some templates have an almost one-to-one correspondence to the rules of the prototype chunker from Chapter 3, e.g. the if-statement template from Figure 5.19d. In other cases the correspondence is more diluted. For instance, the clausal coordination rule (§3.2.1) spans templates such as the one for *and* clausal coordination from Fig. 5.19a, but also includes its variants with separate targets for index and handle links. Since the new chunking



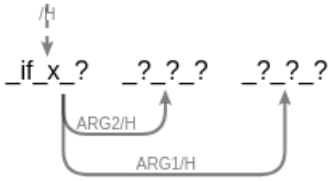
(a) Template for the *and* coordination between simple finite clauses.



(b) Template for the *in order to* control construction.



(c) Template for the VP *and* coordination.



(d) Template for the *if* statement.

Fig. 5.19 Examples of the most common templates.

	Training	Test
Attempted examples	3229	3199
Chunking points	2056	2002
Training templates used	789	303
New templates found	—	459

Table 5.3 Comparison of the templates found in the training and test datasets.

algorithm obeys the scopal hierarchy, it avoids chunking based on coordination inside clausal complements, which proved problematic in earlier experiments (§3.5).

The training and test results are summarised in Table 5.3. The test set comprises 3465 examples, with 3199 meeting the domain requirements. Out of these, 1279 DMRS graphs comprise more than one chunk. 71% of the chunking points are accounted for by 303 templates from the training set, showing strong performance of our generalization principles. Most of the new templates belong to the single-use category, which is also predominant in the training set, with 459 new templates needed to describe 572 chunking decisions unaccounted for by the training set. This is an expected behaviour as most language phenomena follow a long-tail distribution.

5.4.4 Surface templates

Assuming we can realize all the semantic chunks found in §5.2, their surface forms still have to be combined into a full sentence. The information about how the chunks are connected semantically is stored in functional graphs associated with chunking decisions, but the graphs are not well-formed DMRSs and cannot be processed directly by a generator. In §3.6 we obtained their surface representation through the use of placeholders occupying relevant arguments slots. The technique worked well for the restricted form of chunks, but does not scale well with the fewer constraints of the scope-based chunking model.

Although automated, chunking decisions made by the new model can be summarised with underspecified templates (§5.3) that allow us to generalize across the dataset. This means we can not only inspect the chunking decisions more easily but also leverage patterns in the data. In particular, each template is associated with a limited set of syntactic constructions. For example, the template associated with the clausal *and* coordination corresponds to a string pattern *A and B*, where *A* and *B* are the surface representations of the coordinates. In this section we show how the patterns, which we call **surface templates**, can be extracted from training data.

DMRS samples

Most large DMRS examples are made up of more than two chunks. The larger the DMRS, the higher the complexity faced by a processor and the more likely we are to produce a sub-optimal result. In order to decouple templates present in one DMRS and to minimise processing noise, we produce surface templates based on **samples** of DMRS graphs in which they appear.

A DMRS has to meet certain well-formedness requirements to realize successfully. Template graphs, or even full functional graphs, generally do not meet them. A DMRS sample for a given template comprises the source DMRS equivalents of all nodes represented in the template. If the directly equivalent node is a scopal operator, we instead sample its central node unless it is explicitly present in the template. If the operator modifies the grammatical properties of its subgraph (cf. modal verbs in §5.4.1), we change the central node appropriately.

Most nodes can be safely separated from their modifiers, but removing arguments risks breaking the well-formedness conditions. To avoid this, each node is sampled together with its arguments. Additionally, each instance node (§2.3.1) is added to the sample DMRS with its quantifier.

For simple examples the sample ends up an exact copy of the original DMRS, but most of the time the resulting DMRS is much smaller. For the training set introduced in §5.4.3, sampling failed 149 times (10%), largely because of interactions between satellites produced by different templates.

Extracting surface templates

Each sample DMRS matches one template. For each chunking point in the training set, we

1. create a sample DMRS (§5.4.4),
2. realize it,
3. chunk it (§5.2),
4. realize its nucleus and satellite chunks (§5.4.1),
5. find the chunk surfaces in the realization from Step 2 and replace them with corresponding tags, e.g. *[NUCLEUS] if [SATELLITE].* and *If [SATELLITE], [NUCLEUS]* for the template in Fig. 5.19d.

The resulting strings with nucleus-satellite slots become surface templates for other examples matching the template.

English mostly follows a left-to-right ordering, where surfaces of chunks can be arranged according to their in-order position within the tree-shaped scopal hierarchy (§5.1). The nucleus starts the sentence, followed by its closest satellite introduced by the associated operator. The last element on the right corresponds to the rightmost leaf of the scopal tree. The left-to-right arrangement is not universal, even though usually grammatical in English. Semantic chunking is not a task limited to English, and even within that domain, the ordering of main and subordinate clauses is commonly inverted for discourse purposes. The results of realization with chunking should include both variants when possible. Sampling a wide variety of training examples increases the chance of capturing multiple possible patterns.

As expected, not all chunks that fit the requirements discussed in §5.2 can be used for realization. For example, some main clause chunks do not realize without their compulsory clausal complements. Instead of choosing the appropriate templates manually, we can filter them based on the failed and successful training examples.

The procedure outlined above can fail in multiple ways. Either nucleus or satellite might not be suitable chunks for realization, as in the case of compulsory clausal complements. In that scenario, none of the samples based on the template yield a surface template, and the pattern should be ignored for realization purposes. Another cause of failure can be that

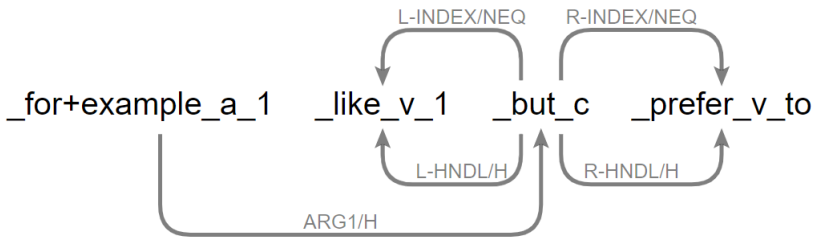


Fig. 5.20 A simplified DMRS for *For example, Merry likes beer but Pippin prefers cider.* Only relevant nodes are shown.

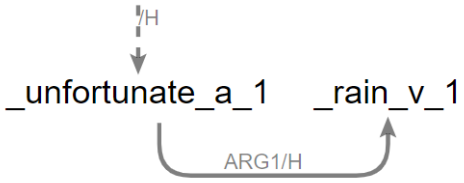


Fig. 5.21 DMRS used to realize the sentential modifier *unfortunately*.

our simple sampling assumptions are not always met. Samples sometimes fail to generate, whether because they are not well-formed or because the original DMRS graphs do not generate either. For a template occurring multiple times in the dataset, only one sample has to succeed for a surface template to be produced. In our experiments we retain only surface templates repeated for at least 30% of occurrences.

When designing nonterminals for realization (§5.4.2), we chose to delexicalise modifiers of operators to limit the number of possible templates. As a result, surface templates explicitly represent operators common for all occurrences of the template, such as *if* or *and*, but the additional sentential modifiers should be realized separately. *For example* in Figure 5.20 is represented in a template as a non-terminal `_?_a_?`, so that the same template can be used for similar sentences modified by other phrases and adverbs with the ERG part-of-speech tag `a`, e.g. *unfortunately*. Otherwise, a very large number of templates would be required to account for all possible adverb lemmas. We realized the modifiers with a placeholder DMRS of *it rained* (e.g. Figure 5.21) and removed the known part of the surface in post-processing. The equivalent string in the realization of the full sample DMRS is replaced with a *[MOD]* string placeholder, e.g. *[MOD], if [SATELLITE], [NUCLEUS].* We use the same method to realize the respective modifiers during realization with chunking. Examples with modifiers applied to multiple nodes are ignored.

Reason	Count
Failed sampling	131
Failed sample realization	360
Failed nucleus realization	548
Failed satellite realization	183
Failed substitution	41
Failed modifier realization	33

Table 5.4 Reasons for failed template realization.

Results

397 out of 789 templates always failed realization. A single attempt can fail in multiple ways (Table 5.4). The most common cause is the nucleus chunk not realizing, which is expected as it happens for all predicates with compulsory clausal complements. Satellites sometimes fail to realize because of errors in the original DMRS graphs or because they are in subjunctive or imperative moods, which do not realize outside their context. Since such chunks do not obey the self-containment constraints for realization, the associated templates are candidates for filtering out.

Some failed modifier realizations occur when multiple independent modifiers are present in a single template. This is space for future improvement. Another common reason is a single modal modifier scoping over both chunks, e.g.

(99) People can create new words and be understood.

(100) `can(and(create(people, new words), be_understood(people)))`

The modal operator node in the DMRS does not interact with the arguments of the main verb, but unlike other shared modifiers, it does interfere with the surface string of one of the chunks. It could be possible to treat such modifiers similarly to shared arguments (§5.1.2) by assigning them to the nucleus chunk and including them in the functional graph. However, such sentences should arguably not be chunked for realization because their chunks are not sufficiently independent.

5.4.5 Assembling the full surface

If all templates for a given example have a surface representation, as found in the previous section, the assembly of the full surface of the sentence is straightforward. Each template is represented as a string with a *[NUCLEUS]*, *[SATELLITE]*, and maybe a *[MOD]* placeholder. Let us consider the example:

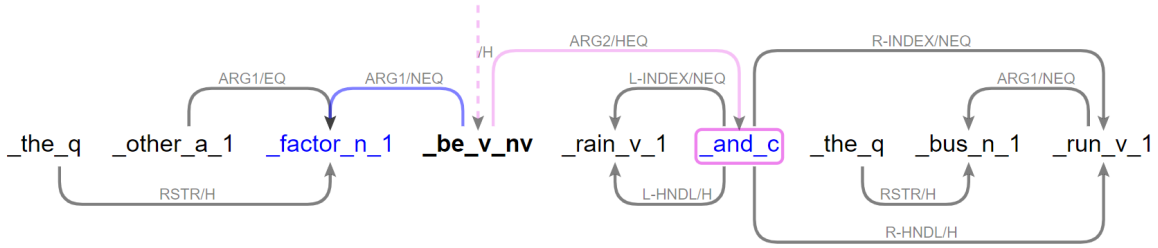


Fig. 5.22 DMRS of Example 102. Some nodes were omitted for better readability.

(101) If it is raining, I don't want to go outside, but I would gladly have a nap.

It comprises three chunks which correspond to:

a) it is raining,

b) I don't want to go outside,

c) I would gladly have a nap,

arranged according to two templates with surface representations *If [SATELLITE], [NUCLEUS]* and *[NUCLEUS] but [SATELLITE]*. The surface templates can be nested, following the scopal relations between chunks, to form the surface template for the entire sentence: *If CHUNK-A, (CHUNK-B but CHUNK-C)*. Once individual chunks are realized, we can substitute their surface representations into appropriate slots.

However, not all templates have a surface representation based on the training set, whether due to insufficient data or because the template is not well-suited for realization (§5.4.3). If a template does not have a surface equivalent, we ignore the boundaries between semantic chunks which it introduces.

Let us assume that the chunks separated by the *but* coordination in the example above are blocked in this way. If they cannot be further chunked themselves, they are never separated and treated as a single larger chunk. Complications occur when one of the disallowed chunks can benefit from splitting into smaller parts. In theory, we can distinguish three chunks in Example 102 (Fig. 5.22), but the nucleus is not a valid standalone chunk for the purpose of realization, as it cannot be realized without its compulsory ARG1/HEQ argument.

(102) [The other factors were] that [it was raining] and [the bus was late].

The argument itself is a complex clause, comprising two chunks connected by a coordination template. The simplest solution would be to ignore any chunks in the scopal trees of the nucleus and satellite connected by the invalid template, but that limits the number of examples which can benefit from chunking. Our example would be left untouched. If we

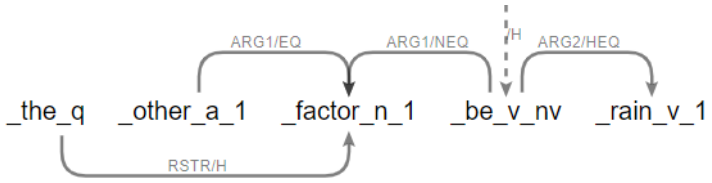


Fig. 5.23 DMRS produced by merging the nucleus of Example 102 and the nucleus of its satellite.

insist on using the partial chunking, the nucleus of the problematic template can only be realized if the satellite role is filled.

The smallest existing subgraph guaranteed to meet the grammatical requirements is the nucleus of the original satellite chunk (*it was raining* in Example 102), provided it can exist without its own satellites. Unfortunately, the semantic subgraph of the main nucleus and of the satellite nucleus are not actually connected in the original DMRS; instead, they interact via the top node of the full satellite, namely the operator *and*. Thanks to the functional graph produced during chunking and because of the scopal properties of DMRS, we can isolate and transpose the interaction onto the desired chunk pair (Fig. 5.23).

Ideally, we could realize the enhanced nucleus, identify the part of the surface produced by the nucleus of the satellite, and use that string to assemble the full surface of the satellite. For our running example, we would know that *it was raining* in *the other factors were that it was raining* is the satellite part, and we could use it as the nucleus in the *[NUCLEUS] and [SATELLITE]* surface template of the satellite after realizing the *[SATELLITE]* part (*the bus was late*). This could be achieved if the generator output contained information about which nodes in the input DMRS were the basis of particular string spans in the output, similar but reverse to the character spans available for parsing (2.3.1).

Alternatively, we could obtain the surface form of the satellite nucleus by realizing it on its own as well as together with the nucleus. That would mean processing it twice, which runs against the spirit of the task, especially in cases where the chunk in question is large. Sampling the required chunk first (§5.4.4) could reduce the cost but at the risk of introducing errors.

In our experiments we decided to fall back onto the left-to-right assumption discussed in §5.4.4, rather than to modify the ACE output to include the information about which nodes produced which tokens. Templates generally allow the left-to-right ordering of their nucleus and satellite, even if the reverse is also possible. Earlier, we established the full surface template of Example 101 to be *If CHUNK-A, CHUNK-B but CHUNK-C*. If all the templates in Example 102 had surface representations, we assume that the full template would be *CHUNK-A CHUNK-B and CHUNK-C*, with the satellite realized to the right of the

nucleus. Since CHUNK-A (*the other factors were*) cannot be realized on its own, we merge its subgraph with that of CHUNK-B – the nucleus of the satellite. The surface realization of the merged subgraph would fill in *CHUNK-A CHUNK-B* part of the full template, based on the left-to-right assumption, while CHUNK-C could be realized on its own. The two results assembled according to the full surface template yield the surface representation of the full sentence.

5.5 Realization results

Realization with chunking using the scope-based chunking model produced surface representations of 763 out of 953 chunked DMRS from the training set. The remaining examples were not chunked when only templates with surface representations were allowed. For 79 of these, no full realization could be produced within the time limit of 40 seconds for 79 of these. In our experimental set up we did not allow partial successful realizations in which not all chunk surfaces can be realized within the timeout. It should be noted, however, that realization with chunking is able to produce partial results for some of the examples where full realization fails, provided that some of the chunks are realized successfully.

	Training	Test
Chunked DMRS	953	935
Realized from chunks	763 (80%)	698 (75%)
Same or better BLEU score	55%	49 %
Average BLEU ($\pm$ std.)	-0.03 ± 0.15	-0.06 ± 0.16

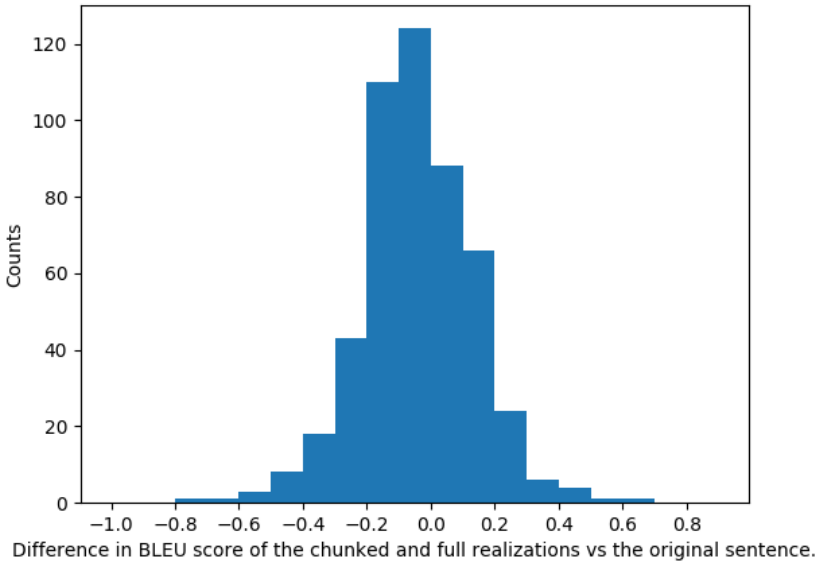
Table 5.5 Results of chunked realization for training and test sets

During the experiments on the test set, we allowed only chunks based on templates from the training set which have surface representations (cf. §5.4.3). There are 935 such examples. Out of these, 698 were realized from their semantic chunks. For 42 examples, this was the only realization we could produce within the constraints. The comparison of the numbers for the two datasets can be seen in Table 5.5. Overall, we observe a small but expected decrease in the number of successfully processed examples for the test set. Since we restricted chunking only to the templates from the training set, more examples in the test set contain more complex chunks, which would potentially be further reduced if a matching template was available.

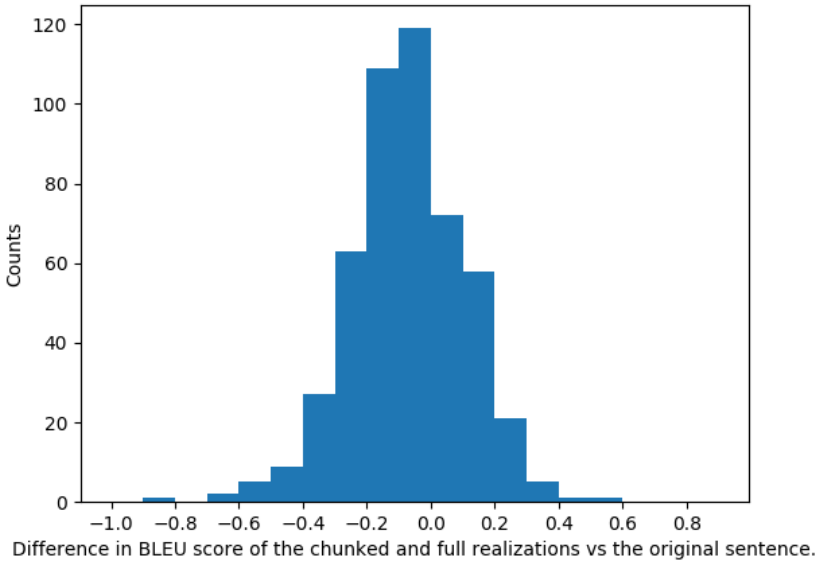
We evaluated the top result for full realization and chunked realization against the original sentence, using sentence-level BLEU score. Before comparison, all strings were converted to lower case, stripped of punctuation and tokenized. We use the NLTK toolkit (Loper and

Bird, 2002) for both tokenization and BLEU score calculation. The plots in Figure 5.24 show the difference in the BLEU score against the original surface obtained by the top chunked realization and the full realization. The top chunked realization comprises top realization results of chunk subgraphs and the most common template representations. The histogram does not include ties. Overall, the chunked realization produced the surface of the same quality or better than full realization 55% time in the training set and 49% in the test set. The average BLEU score difference is -0.03 ± 0.15 and -0.06 ± 0.16 respectively. The average BLEU score for the test set is is virtually indistinguishable from the score for the training set, well within a fraction of its standard deviation, supporting the generalisations underlying our experiments. This demonstrates yet again that processing DMRS graphs in chunks does not affect the quality of results. Furthermore, it shows that the quality persists even after adding a wider variety of chunked structures available in the new chunking model (cf. §3.6.3).

The average BLEU score for the examples which did not generate in full is 0.70 ± 0.16. Improvements could be achieved if the chunked realizations were ranked using a model trained specifically for realization with chunking (§3.6.3).

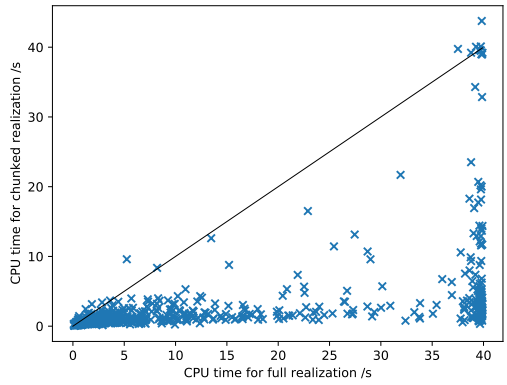


(a) Training

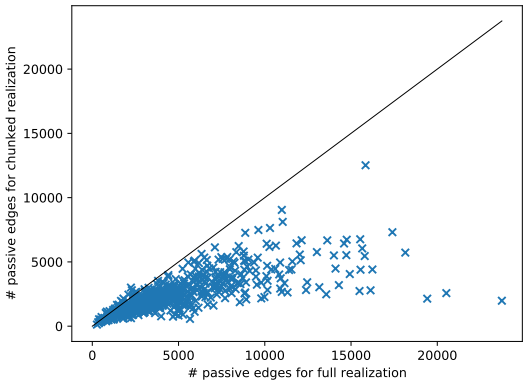


(b) Test

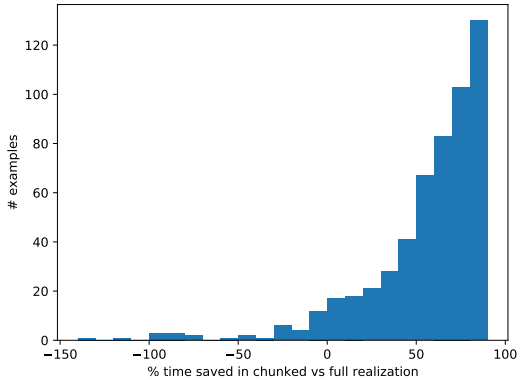
Fig. 5.24 The difference in BLEU scores of full realization and realization with chunking against the original sentence. Positive value indicates that the chunked result scored higher.



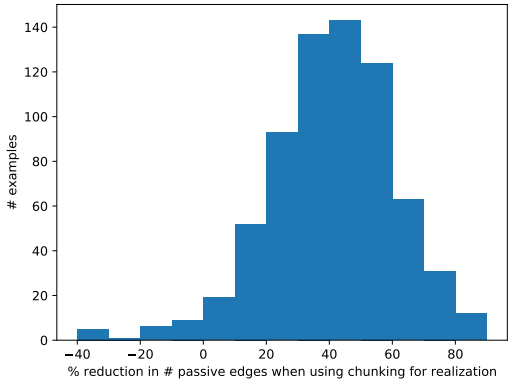
(a) Time.



(b) Passive edges.



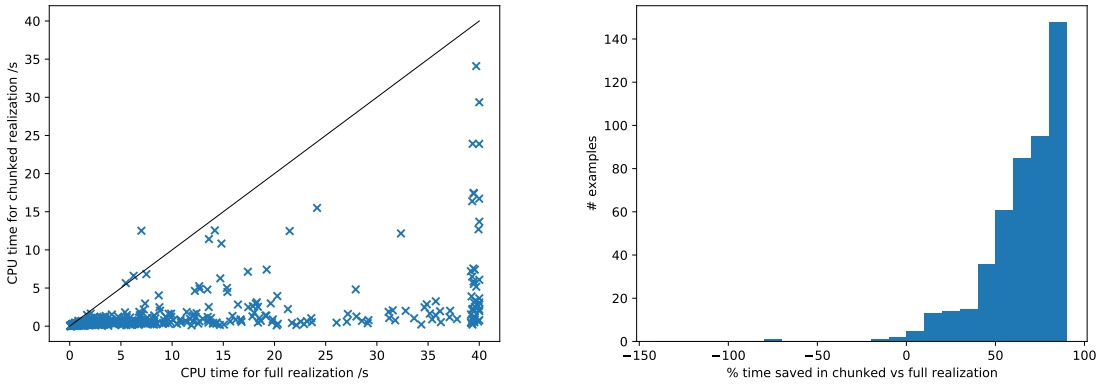
(c) Percentage improvement in time.



(d) Percentage improvement in the number of passive edges.

Fig. 5.25 Comparison of resources needed for the realization of full examples in the training set and for their realization with chunking.

The scatter plots in Figure 5.25 show the CPU time taken to produce the realizations of training examples as reported by ACE, and the number of passive edges constructed in the process. The number of passive edges is a system-independent metric of the complexity of chart generation. For chunked realization, the values used are the sums of times/edge counts required by realizations of various subgraphs. Since individual chunks can be realized in parallel, this is a conservative measure of performance gains. Both plots echo strongly the equivalent figures from §3.6.3. In the training set 106 examples needed more than 39 seconds to produce the full result, cutting the computation short due to timeout. Similar time was needed by 6 chunks. The overhead due to chunking and reconstruction is negligible, compared with an average time required for realization, while the cost of template realization should be considered as amortized against test time.



(a) Time.

(b) Percentage improvement in time.

Fig. 5.26 Comparison of resources needed for the realization of full examples in the test set and for their realization with chunking.

When it comes to the number of passive edges produced (Fig. 5.25b, only 20 examples in the training set required more or the same number of edges in the chunked realization as in the full one. These tend to require a low number of edges in the first place, which agrees with the intuition that smaller examples do not benefit as much from chunking as more complex ones.

Figures 5.25c and 5.25d show the same data for the training set as the plots above them, but they highlight the percentage improvements in the performance metrics that were achieved by the introduction of chunking. The negative x-axis values correspond to the examples for which realization with chunking required more resources than the standard approach, i.e. the outliers in the scatter plots. On average, the introduction of chunking reduced the time of realization by 63% and the number of passive edges by 41%[4].

We present the improvements in the time required for the realization of the test set in Figure 5.26. The improvements in the time required for realization persist, as the plots show the same pattern as their training set equivalents. In fact, the test set leads to fewer outliers. We hypothesise that this difference is due to the absence of unique chunking decisions in the test set, since the test examples were chunked only using the templates already present in the training data.

[4]Two-tailed Wilcoxon signed-rank test (Wilcoxon, 1945), $p < 0.001$ (cf. §3.6.3).

5.6 Conclusions

The chunking system described in this chapter uses the scopal properties of DMRS to automatically identify semantic subgraphs suitable to act as semantic chunks. The inherent structure of DMRS allows us to depart from hand-written rules, which do not suit the flexible nature of semantic chunking as a task. As a result, we were able to investigate chunks based on a broader variety of constructions. The realization experiments from §5.4 show that despite the precise nature of the task, the relaxation of constraints did not impact the quality and performance of realization with chunking, both as compared with the realization from full sentences and with the realization using the prototype chunking model from Chapter 3.

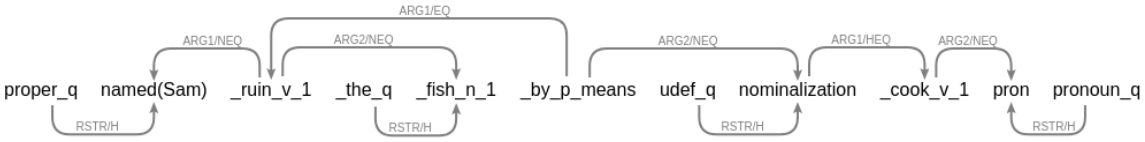
The automated approach does not mean that we need to give up control over tailoring semantic chunks to the needs of the target task. The new chunker replaces prescriptive rules of the prototype with descriptive templates, which are underspecified representations of chunking patterns, generalized over similar examples. The templates allow us to leverage training data for the purpose of a) filtering the suitable chunks, b) opening up new methods of processing the chunked examples. In §5.4 we returned to the task of realization, which we previously investigated in §3.6. The changes in the approach to realization with chunking include not only new types of semantic chunks, but also a new way of creating surface representations of functional graphs. As shown in §5.4.4, each DMRS template has a potential matching surface template, which can form a scaffolding for the assembly of realizations of its associated chunks into the surface of a full sentence.

This chapter concludes our practical investigation into semantic chunking. We continue our discussion in the next chapter, where we summarise the findings and suggest future research directions. Before that, however, we would like to mention one potential research topic, which is directly related to the contents of this chapter.

5.7 Further work: Non-scopal chunks

The chunking approach based on scopal relationships reflects inherent properties of *MRS representation, but does not capture all constructions capable of acting as semantic chunks. Scopal arguments match the concept of situation discussed in Chapter 2 well enough that the presence of scopal links allowed us to expand viable chunk centres to parts of speech other than verbs. It reliably indicates non-attributive occurrences of adjectives and prepositional phrases. In contrast, when considering non-scopal chunks, there are no clear criteria on which to base the distinction. Verbs and tensed nodes seem to be the only safe candidates for chunk centres.

(a) DMRS of *the cat which sat on the roof.*



(b) DMRS of *Sam ruined the fish by cooking it.*

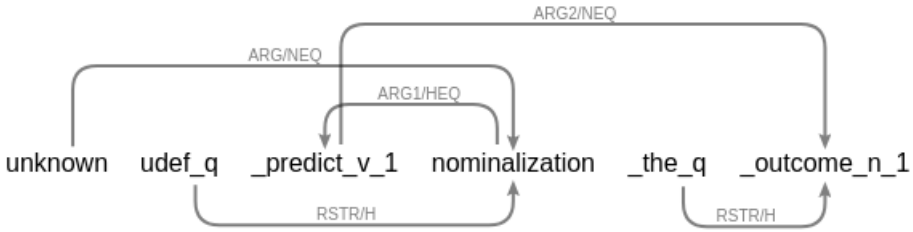


Fig. 5.28 DMRS of *predicting of the outcome/predicting the outcome.*

In particular, two major categories of chunks elude detection based on scopal relationships: relative clauses (Fig. 5.27a) and gerund clauses introduced by prepositions (Fig. 5.27b). Central nodes of these satellites have no scopal connection to the top of the sentence and because of that, they are not captured by the scopal hierarchy introduced in §5.1.

Relative clauses are parts of nominal phrases, with the antecedent of the clause acting as the top node of their subgraph. Prepositional phrases are themselves modifiers, with verbs bound within a nominalized phrase by a `nominalization` predicate. Since the ERG 1214 does not maintain a clear distinction between gerundive and mixed nominals (§2.1.3), it is unclear whether subgraphs introduced by `nominalization` nodes should be considered valid chunks. The DMRS graphs from Figure 5.28 generates both of the following surface forms:

(103) predicting the outcome,

(104) predicting of the outcome.

The ambiguity is amplified by the fact that the same type of nominalized phrase can be used in coordination with nominal phrases based on nouns and as sentence subjects.

Given the constraints on valid chunk centres, grammatical constructions with non-scopal chunks can be represented as templates, just as their scopal counterparts. The starting point would be a DMRS graph of a scopal chunk, which is the nucleus situation, e.g *Sam ruined the fish* in Figure 5.27b. The functional graph for the chunking decision would comprise nodes participating in simple paths between the two chunk centres, and could be converted to a template, like other functional graphs we encountered. Long prepositional phrases with gerund arguments act syntactically like subordinate clauses, falling naturally to the end of the sentence, so we expect the templates to produce usable surface representations, similar to those of scopal templates.

Relative clauses pose more problems, as they can modify any element of the chunk, independent of its syntactic position in the sentence. For example, *the cat* from Figure 5.27a could be the subject or the object of the nucleus clause:

(105) The cat which sat on the roof looked bored.

(106) I miss the cat which sat on the roof.

In Example 105 the relative clause is interjected in the middle of the surface string representing the nucleus chunk. There are no heuristics similar to the left-to-right assumption which we used for scopal chunks. If we had access to the mapping between realized tokens and their source nodes, we could replace the string of the antecedent in the realization of the nucleus with the instantiated template representation, using surface representations of templates such as *ANTECEDENT which SATELLITE*. Without that information, assembling the full surface string out of the resulting chunks is not straightforward. The alternative solution is to realize the entire template and identify the common substring between the nucleus surface and the template realization as the antecedent string. The downside of this approach is that the antecedent would have to be realized twice, potentially incurring significant computational cost.

Addressing these difficulties would allow the chunking model described in this chapter to include an even wider variety of semantic chunks.

Chapter 6

Conclusions

This thesis introduced a new task called semantic chunking, dedicated to reducing processing costs of downstream NLP tasks reliant on semantic information. In this chapter we give an overview of its contributions, expanding upon the list from §1.2, and we point out potential future research directions for each aspect of the investigation.

6.1 The foundations of the task

The main contribution of our work is the introduction of semantic chunking. The starting point was a practical definition of the task designed to isolate semantically contained fragments of a sentence that could be processed independently without loss of information. We envisioned a pre-processing divide-and-conquer approach tailored to the needs of the input representation and the target task. The results of processing individual fragments, i.e. semantic chunks, can be assembled into a result that is identical to the output of the target task performed on the full sentence, but which was obtained at a smaller computational cost.

Semantic chunking defined in this way relates to existing NLP tasks, such as shallow chunking and sentence simplification (§2.2), especially the ones that act as pre-processing steps for applications requiring semantic annotation, e.g. machine translation or summarisation. The distinguishing feature of semantic chunking is the flexibility of what constitutes a valid output, and its primary consideration is the practical improvement observed in the downstream task. The customisable nature of semantic chunking affects the interpretation of the condition that semantic chunks have to be semantically contained. This constraint has to be resolved in the context of the application and its inputs. If an aspect of semantic information is irrelevant to the task or inaccessible through the given representation, semantic chunking does not have to preserve it. At the same time, auxiliary structures can allow chunks

to interact by preserving the information that would otherwise be lost if the constituents in question are separated (cf. §6.2).

In a sense, many formulations of the related tasks can be considered specific cases of semantic chunking. For example, one of the common steps in sentence simplification (§2.2.2) is splitting a sentence into smaller fragments and the elements of the models responsible for this operation can be seen as performing a dedicated form of semantic chunking. The task defined in this thesis is in a way a generalization of the similar tasks.

In Chapter 2 we investigated theoretical concepts used to describe the type of information that semantic chunking is interested in preserving. After exploring propositions as potential theoretical equivalents of semantic chunks (§2.1.1), we homed in on situation and event semantics as two related frameworks which offer the most insights into the form of suitable constituents (§2.1.2). Situations combine propositions that share participants and spatiotemporal locations and, together with event variables, establish the basis for distinguishing semantically contained constituents.

The task is deeply rooted in the principle of compositionality, reflected in the underpinning assumption that a sentence can be divided into constituents suitable to act as semantic chunks. In §2.1.3 we identified individual situations with syntactic clauses in a sentence, reviewing a variety of other relevant grammatical constructions, such as eventive noun phrases. In the process, the issue of granularity emerged as a key consideration. Semantic chunks have to be small enough to be processed more readily than their full source sentences, but at the same time they have to be large enough to be distinct from individual units of representation, e.g. tokens in a sentence string. This balancing act is one of the examples how semantic chunking is defined primarily through its practical goals and how its exact form depends on the intended use.

Identifying semantic chunks as related to situations and syntactic clauses provided us with a framework in which we can consider relationships between chunks (§2.1.4). We borrow terminology from Rhetorical Structure Theory (RST) to connect semantic chunks within an individual sentence through nucleus-satellite relationships, as we observe that each sentence has a nucleus semantic chunk describing the core situation expressed by a sentence. The nucleus chunk is further qualified by satellite chunks. Syntactically, the relationships are often expressed by operators, e.g. subordinating conjunctions, which do not belong to semantic chunks but have no independent meaning. Throughout the thesis we referred to them as functional fragments and investigated ways in which the information they convey can be preserved during chunking for the purpose of the final assembly step of partial results into the full output. We summarise the explored solutions in §6.3.

The primary representation on which we focused in the thesis is Dependency Minimal Recursion Semantics (DMRS), which takes the form of semantic dependency graphs. We gave an overview of DMRS and its related *MRS framework in §2.3. *MRS is a formalism with a deeply in-built syntax-semantics interaction and a sophisticated treatment of scopal relationships. Both of these properties make it an appealing domain for the exploration of semantic chunking. On one hand, strong compositional properties of the representation, supported by a wide coverage, linguistically motivated grammar, mean that constituents suitable to act as semantic chunks emerge naturally from DMRS structures (cf. Chapter 5.3). On the other hand, the close knit relation of the complex representation to the underlying surface form creates an opening into the investigation of semantic chunking of the surface string representation (§6.2, cf. Chapter 4).

6.1.1 Further research on the foundations

Our thesis practically explored semantic chunking on two representations: DMRS graphs and surface strings, but the task could be defined on other ones as well. The experiments of Narayan and Gardent (2014) on sentence simplification (§2.2.2) suggest that semantic chunking could be successfully applied to Discourse Representation Structures (DRS; Kamp (1981)). Another promising formalism is Abstract Meaning Representation (AMR; Banarescu et al. (2013)), which is a representation language, rather than a fixed representation itself. Like DMRS, AMR structures are graphs with directed edges representing relations, so that some of the techniques developed in this thesis could be adapted to serve the new representation. On the other hand, like DRS, the information contained in AMR graphs is dependent on the particular annotation conventions adopted and often encompasses elements of discourse. Both DRS and AMR could open a way to researching a chunking-like task which is not limited to the semantic information of individual sentences but can target larger text spans.

Another expansion of the task domain could see chunking applied to different languages. The thesis focuses exclusively on English, but the DELPH-IN initiative (§2.3) originated *MRS-based grammars of multiple languages, including Japanese, German and Spanish. These grammars could be used to assess how well our observations transfer to other languages.

Finally, the related task of speech segmentation (§2.2.4) suggests another potential extension. Like semantic chunks, speech units aim to divide an utterance into semantic fragments that facilitate the processing of longer input. Although idiosyncrasies of speech mean that spoken utterances fall outside of the scope of this thesis, semantic chunking could be adapted to meet the needs of that domain.

6.2 Chunking models

After establishing the theoretical foundations of the task in Chapter 2, we proceeded to investigate how semantic chunking and its underlying principles can be implemented in practice. The key contribution of the core chapters is the introduction of three chunking approaches: two dedicated to chunking DMRS representations, and one which models surface semantic chunks directly based on semi-supervised training data.

The chunking system described in Chapter 3 performs semantic chunking on DMRS graphs following a set of hand-crafted rules which target a selected range of grammatical constructions. Its focus was to demonstrate the viability and benefits of semantic chunking as a pre-processing step for realization from DMRS. In the *MRS framework the operation can be thought of as the inverse of parsing and it requires a well-formed input. With that in mind, we prioritised quality of chunks over their quantity and restricted their form to finite clauses, which are readily processed by the generator (§3.1). The rules (§3.2) reflect grammatical constructions which combine multiple such clauses and which can be reliably identified locally in DMRS graphs: clausal coordination, subordinating conjunctions, and clausal complements of verbs.

Another reason behind the focus on the quality of semantic chunking over wide coverage was the secondary objective of the proof-of-concept chunker, which is related to the second chunking model introduced in the thesis. Chapter 4 describes how DMRS semantic chunks can be used to train a surface-based chunking model. Using the connection between DMRS graphs and the strings from which they were created, we successfully converted semantic chunks defined on DMRS into surface fragments suitable to act as surface semantic chunks (§4.1). Guided by the state-of-the-art approaches to similar tasks, we trained a semi-supervised sequence labelling model capable of chunking previously unseen examples without access to their semantic representations (§4.3). The chunker reached the F-score of 0.862, below but comparable to the scores for established sequence tagging tasks such as NER or shallow chunking. The size of targeted fragments, complex nature of captured dependencies, and automatically created dataset make the task more challenging than some of the existing ones, but taking into account the lack of in-depth hyperparameter optimization, the score suggests that the sequence labelling approach is suitable for the task.

The hand-crafted rules of the prototype model covered only a small range of phenomena discussed theoretically in Chapter 2 as potential bases for valid semantic chunks. This resulted in rare chunking opportunities and long, sub-optimally complex chunks. Our experiments exposed other brittle aspects of the rule-based approach, such as difficulties capturing detailed interactions between constructions (§3.5). Although sufficient to demonstrate the benefits of chunking, the human-centred design stood at odds with the flexible philosophy of the

task. In Chapter 5 we set out to explore to what extent the restrictions on the form of chunks can be loosened without reducing the performance gains and the quality of results (cf. §6.3; §§3.6.3, 5.5). The new chunking model steps away from prescriptive rules and instead relies on suitable constituents emerging naturally from the grammar principles reflected in the DMRS structure. The key feature behind its operation is the scopal hierarchy of DMRS, providing a tree-like organisation of subgraphs corresponding to individual situations. In §5.1 we demonstrate how a simple set of constraints on the form of chunks leads to coherent chunking decisions in a multitude of grammatically complex scenarios. For example, the expanded catalogue of semantic chunks includes fragments with shared entities, as in the case of VP coordination.

The result is an automated chunking model capable of finding chunking opportunities based on previously unseen DMRS structures. The human input is limited to defining constraints on what constitutes a valid semantic chunk for the target downstream application. Some of them are defined a priori, but the results can be further refined by applying filtering criteria to the chunking opportunities discovered by the model. The patterns in chunking decisions can be generalized in the form of templates (§5.3), i.e. underspecified DMRS graphs which summarise chunking decisions by abstracting them from irrelevant lexical details. Template-based generalisations can also open up new ways of processing chunked sentences, as we demonstrate with surface templates for realization (§5.4.4; cf. §6.3).

6.2.1 Further research on chunking models

One potential improvement to the models presented in this thesis has already been suggested in §5.7. The scopal framework of the chunker from Chapter 5.3 can be readily expanded to include non-scopal chunking opportunities, such as relative clauses.

Throughout the thesis, we pointed out granularity as one of the parameters determining the suitable form of chunks for a given task. We observed that the long and complex semantic chunks produced by the rule-based chunker could be behind a lower performance of sequence labelling on semantic chunking when compared with other similarly framed tasks. On the other hand, the scope-based system of Chapter 5 was designed to explore the boundaries of what constitutes a valid chunk under minimal assumptions and does not impose direct limits on the chunk size. It would be interesting to investigate in-depth and quantitatively how the size of semantic chunks affects the performance and quality of the results of downstream tasks, and whether it is possible to establish a principled optimal threshold. It would have to determined on a per target task basis and take into account its particular implementation, as the optimal chunk size is likely different for a chart generator, e.g. ACE, than for a neural model, e.g. Hajdik et al. (2019).

Just as the DMRS chunking model from Chapter 3 was the proof-of-concept for the task in general, the surface chunking model from Chapter 4 aimed to show that it is possible to model surface semantic chunks directly from sentence strings. Although our approach was guided by state-of-the-art models successful on other sequence labelling tasks, we did not focus on in-depth optimization. We find it likely that the task score could be raised further by careful hyperparameter optimization and tweaks in the details of the model architecture. The recent developments of contextual word embeddings, such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), further pushed the limits of benchmark tasks such as NER, and it would be interesting to explore how they interact with the semantically oriented objective of semantic chunking.

As much as surface semantic chunking resembles other sequence labelling problems, it has some unique features which were not addressed in our experiments. In particular, semantic chunks are intrinsically nested structures and because of that, chunking decisions can be thought of as segmentations. This characteristic is particularly exposed by the scope-based model, where each new chunk is the result of deciding whether a satellite clause meets the criteria for a valid chunk. If we were to impose a minimum threshold on the size of chunks, some large fragments comprising multiple clauses would remain whole as large semantic chunks, even though a lower threshold could lead to their division into smaller chunks. The problem of nested sequence labelling has drawn less attention in the general sequence labelling research because of the common flat formulations of sequence labelling benchmark tasks. Nested NER (Finkel and Manning, 2009) is, however, a busy research area with a variety of actively investigated approaches to the problem. Some of the recent ones include linearization (Straková et al., 2019) and stacking of multiple flat models (Ju et al., 2018). The nested nature of semantic chunks would make the task another target of these architectures.

In a natural next step we would like to investigate how well the more varied DMRS chunks of the scopal system translate to surface chunks and what constraints are suitable for the target task of parsing (cf. §6.3.1). The limited set of hand-crafted rules behind the original model allowed a straightforward conversion into a labelling scheme. The wider variety of modelled constructions associated with the new chunker would require a more sophisticated set of tags. Ideally, we would be able to recover the information about which template corresponds to each chunking decision, so that the target task could benefit from the template structure when combining partial chunk results into the full analysis. We are confident that the general nature of templates lends itself to conversion into an informative labelling scheme based partially on nonterminals describing each chunk. The more complex distinctions could be modelled by adding a complementary classifier layer on top of the

sequence tagger, with the tagger responsible for locating chunk boundaries and the classifier linking them to one of the templates.

6.3 Processing with semantic chunking

Apart from the three chunking models described in the previous section, the contributions of this thesis include a range of ideas for applying semantic chunking to downstream tasks. The main recurring target we focused on was realization from DMRS, using finite clause chunks of the rule-based chunker and the wider variety offered by the scopal system. The processing paradigm associated with chunking is the divide-and-conquer approach, in which partial results based on semantic chunks are combined in the end step into the output for the full sentence. Both sets of realization experiments (§3.6.3, §5.5) illustrate how the assembled results can match the output of full processing in quality, while offering considerable performance benefits.

The majority of semantic content of the sentence is expressed in the semantic chunks themselves, but the information about how they combine into the full sentence is not included. We store the associated information in functional fragments, which often are not well-formed fragments of representation. In DMRS they are DMRS subgraphs with missing compulsory arguments or even consisting of only links (e.g. for clausal complements), while in surface strings they can correspond to disjoint token spans. The challenge of the assembly step lies in the faithful and efficient inclusion of the information from functional fragments into the final result.

In this thesis we proposed two methods of incorporating the functional information. The functional subgraphs of the rule-based chunking model from Chapter 3 were realized using small placeholder DMRS graphs with known surface representations (§3.6.1). We enhanced the functional graphs with placeholders at their points of contact with semantic chunks, realized the resulting well-formed DMRS and replaced the known placeholder strings with the realization results of actual chunks. The approach worked because of the strong constraint on the form of chunks, as finite clauses are to large extent grammatically interchangeable.

The wider variety of chunks produced by the second DMRS-focused model makes the placeholder approach impractical. Each type of chunk, as defined by its nodes participating in the template, would require a dedicated placeholder. Instead, we put to use the generalization capabilities of the new framework and extracted the representations of functional fragments from the training data into surface templates (§5.4.4), sampling the DMRS graphs in order to decouple multiple templates applicable to the same sentence (§5.4.4). Whether or not

we could identify the surface pattern associated with each template served as an additional criterion on whether the given chunking decision was appropriate for realization.

The final practical contribution of this thesis is the transfer of semantic chunks between representations (§4.1). We successfully demonstrated how a chunking model defined on one representation can be used to bootstrap a system for another one on the example of DMRS graphs and surface strings. As suggested in §6.1.1, semantic chunking of DMRS can also inform similar models using other graph-based representations. The surface chunks produced by the transfer were evaluated using an intrinsic evaluation procedure (§4.2), which is less computationally costly than retraining the full model. To accompany the evaluation, we designed an F-score comparison metric (§4.2.2) for inexact graph matching on DMRS (§4.2.1). The measure also provided a filtering criterion for the training data for the sequence labelling model.

6.3.1 Further research on processing with semantic chunking

The practical solutions to semantic chunking were exemplified in this thesis with the target task of realization. It was chosen because of its close relationship to both types of representation we work with and the demanding requirements of high precision in its inputs. The surface semantic chunks created in Chapter 4, however, remain to be evaluated on the target application of parsing. Other tasks that can benefit from surface chunking include machine translation and summarisation, both particularly dependent on semantic content of their inputs. The DMRS-based models can directly assist existing applications within the *MRS framework, e.g. HSSR/T (§5.3.1) or the summarisation system by Fang et al. (2016).

The difficult question of how to best incorporate the functional fragments into the construction of full results is another area of potential focused research. The processing of the fragments should not incur significant computational costs and should avoid redundancies, e.g. due to processing the same DMRS node multiple times. One of the improvements we suggest is the extraction of predicate-span information from the generator, so that each token in the realization result can be traced back to the node which produced it (cf. §5.4.4). This would allow for a better treatment of sentential modifiers and remove the need for English-centric left-to-right assumptions on which we rely in our experiments. At the same time, the node-token association would potentially open a way for disjoint surface semantic chunks and functional fragments.

The final addition to the experiments could target realization ranking. ACE ranks its outputs with a maximum entropy model trained on treebanked data. In §3.6 we assigned the final output a score by summing the logarithms of scores of individual chunk realizations, while in §5.5 we simply combined the best scoring version of the surface of each chunk.

We expect that training a dedicated model for ranking chunk realizations could improve the quality of the results. Since realization with chunking narrows down the search space of available analyses, a custom ranking model has the potential to outperform the standard approach by selecting optimal partial results.

6.4 Summary

In this thesis we introduced a new task of semantic chunking, designed as a custom and flexible pre-processing step for downstream NLP application. Its main focus is reducing the computational cost of processing complex sentences. We presented three models operating on a complex semantic representation of DMRS and on surface strings of sentences. They were based on theoretical investigation into constituents of semantic meaning and inherent properties of representations. We demonstrated how they can be applied in practice on the example of realization, developing several techniques for preserving and combining all the relevant information. The option of direct chunking of sentence strings makes chunking applicable to more downstream tasks. In particular, it can best benefit tasks which rely on deep semantic interpretation of text, such as machine learning and summarisation.

References

Steven P. Abney. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Springer Netherlands, Dordrecht, 1992.

Roee Aharoni and Yoav Goldberg. Split and rephrase: Better evaluation and stronger baselines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 719–724, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-2114/.

J. L. Austin. Truth. *Aristotelian Society Supplementary Volume*, 24(1):111–128, July 1950.

Emmon Bach. On time, tense and aspect: An essay in English metaphysics. In Peter Cole, editor, *Radical Pragmatics*, pages 62–81. Academic Press, New York, 1981.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-2322/.

Jon Barwise and John Perry. *Situations and Attitudes*. MIT Press, Cambridge, MA, 1983.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. Layers of interpretation: On grammar and compositionality. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 239–249, London, UK, April 2015. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W15-0128.

Endika Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France, Dec 2002.

A. Bies. Bracketing guidelines for Treebank II style — Penn Treebank project. Technical report, Department of Linguistics, University of Pennsylvania, Philadelphia, PA, USA, 1995.

Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. Learning to split and rephrase from Wikipedia edit history. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 732–737, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1080.

Jan Buys and Phil Blunsom. Robust incremental neural semantic graph parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1215–1226, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P17-1112.

Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-2131.

Andrew Caines, Michael McCarthy, and Paula Buttery. Parsing transcripts of speech. In *Proceedings of the Workshop on Speech-Centric Natural Language Processing*, pages 27–36, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W17-4604.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop in Natural Language Generation (ENLG)*, pages 86–95, Toulouse, France, 1999.

Roberto Casati and Achille C. Varzi, editors. *Events*. Dartmouth, Aldershot, 1996.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan, 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P05-1033.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2): 201–228, 2007. URL https://www.aclweb.org/anthology/J07-2003.pdf.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W14-4012.

Noah Chomsky. Remarks on nominalization. In R. Jacobs and P. Rosenbaum, editors, *Readings in English transformational grammar*, pages 184–221. Ginn, Waltham, MA, 1970.

Ann Copestake. *Implementing typed feature structure grammars*, volume 110 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanford, CA, 2002.

Ann Copestake. Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece, March 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E09-1001.

Ann Copestake and Dan Flickinger. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece, May

2000. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L00-1276/.

Ann Copestake, Alex Lascarides, and Dan Flickinger. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France, 2001. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P01-1019.pdf.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics: An introduction. *Research on Language and Computation*, 3(4):281–332, 2005. doi: 10.1007/s11168-006-6327-9.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyńska. Resources for building applications with Dependency Minimal Recursion Semantics. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1240–1247, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1197.pdf.

James R. Curran, Stephen Clark, and Johan Bos. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic, 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P07-2009.

Donald Davidson. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–120. University of Pittsburgh Press, 1967.

Meinou De Vries, Morten Christiansen, and Karl Magnus Petersson. Learning recursion: Multiple nested and crossed dependencies. *Biolinguistics*, 5(1-2):010–035, 2011.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N19-1423.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July 2015. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P15-1033.

Yimai Fang and Simone Teufel. A summariser based on human memory limitations and lexical competition. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 732–741, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E14-1077.

Yimai Fang and Simone Teufel. Improving argument overlap for proposition-based summarisation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–485, Berlin, Germany, August 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P16-2078.

Yimai Fang, Haoyue Zhu, Ewa Muszyńska, Alexander Kuhnle, and Simone Teufel. A proposition-based abstractive summariser. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 567–578, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1055.

Jenny Rose Finkel and Christopher D. Manning. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*, pages 141–150, Singapore, August 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D09-1015.

Dan Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6:15–28, 2000.

Dan Flickinger. Accuracy vs. robustness in grammar engineering. In Emily M. Bender and J. E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage, and Processing*, pages 31–50. CSLI Publications, Stanford, 2012.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. WikiWoods: syntacto-semantic annotation for English Wikipedia. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L10-1296/.

Dan Flickinger, Emily M. Bender, and Stephan Oepen. Towards an encyclopedia of compositional semantics: Documenting the interface of the English Resource Grammar. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 875–881, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L14-1457/.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-2501.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with LSTM. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, September 1999.

Jonathan Ginzburg and Ivan A. Sag. *Interrogative Investigations*, volume 123 of *CSLI Lecture Notes*. Center for the Study of Language and Information (CSLI), Stanford, California, 2000.

Michael Wayne Goodman. A Python library for deep linguistic resources. In *2019 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*, Singapore, October 2019.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M. Bender. Neural text generation from rich semantic representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2259–2266, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N19-1235.

James Higginbotham. On semantics. *Linguistic Inquiry*, 16:547–593, 1985.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California, June 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N16-1162.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

Matic Horvat. *Hierarchical statistical semantic translation and realization*. PhD thesis, University of Cambridge, 2017.

Matic Horvat, Ann Copestake, and Bill Byrne. Hierarchical statistical semantic realization for minimal recursion semantics. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 107–117, London, UK, April 2015. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W15-0116.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv e-prints*, art. arXiv:1508.01991, August 2015. URL https://arxiv.org/abs/1508.01991.

Rodney Huddleston and Goeffrey K. Pullum. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge, 2002.

Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. Chunk-based decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1901–1912, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P17-1174.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1446–1459, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N18-1131.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 2009.

Hans Kamp. A theory of truth and semantic representation. In J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof, editors, *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum, Amsterdam, 1981.

Kate Kearns. *Semantics*. Palgrave Modern Linguistics. Macmillan International Higher Education, 2nd edition, 2011.

Jeffrey C. King. Structured propositions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019. URL https://plato.stanford.edu/archives/sum2019/entries/propositions-structured/.

Jeffrey C. King, Scott Soames, and Jeff Speaks. *New Thinking About Propositions*. Oxford University Press, Oxford, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *Conference Track Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA, May 2015.

Walter Kintsch and Teun A Van Dijk. Toward a model of text comprehension and production. *Psychological review*, 85(5):363, 1978.

Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006. URL https://www.aclweb.org/anthology/J06-4003.pdf.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P03-1054.

Philipp Koehn and Kevin Knight. Feature-rich statistical translation of noun phrases. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 311–318, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P03-1040.

Angelika Kratzer. Facts: Particulars or information units? *Linguistics and Philosophy*, 25 (5-6):655–670, 2002.

Angelika Kratzer. Situations in natural language semantics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019. URL https://plato.stanford.edu/archives/sum2019/entries/situations-semantics/.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N16-1030.

Vilson J. Leffa and Rua Felix Da Cunha. Clause processing in complex sentences. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC '98)*, pages 937–943, Granada, Spain, May 1998.

Joël Legrand and Ronan Collobert. Phrase representations for multiword expressions. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 67–71, Berlin, Germany, August 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W16-1810.

V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

Rochelle Lieber. Nominalization: General overview and theoretical issues. In *Oxford Research Encyclopedias: Linguistics*. Oxford University Press, June 2018. URL https://oxfordre.com/linguistics/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-501. Accessed: March 8, 2020.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D15-1176.

Jan Tore Lønning, Stephan Oepen, Dorothee Beermann, Lars Hellan, John Carroll, Helge Dyvik, Dan Flickinger, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, Victoria Rosén, and Erik Velldal. LOGON. A Norwegian MT effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden, 2004.

Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1 (ETMTNLP '02)*, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P04-3031.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P16-1101.

William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text - Interdisciplinary Journal for the Study of Discourse*, 8 (3):243–281, 1988.

Ryan McDonald and Joakim Nivre. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230, 2011. URL https://www.aclweb.org/anthology/J11-1007.

Matthew McGrath and Devin Frank. Propositions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2018 edition, 2018. URL https://plato.stanford.edu/archives/spr2018/entries/propositions/.

Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28, 1988. URL https://www.aclweb.org/anthology/J88-2003.

Ewa Muszyńska. Graph- and surface-level sentence chunking. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 93–99, Berlin, Germany, 2016. URL http://anthology.aclweb.org/P16-3014.

Ewa Muszyńska and Ann Copestake. Realization of long sentences using chunking. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 218–222, Santiago de Compostela, Spain, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W17-3533.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 807–814, Madison, WI, USA, 2010. Omnipress.

Shashi Narayan and Claire Gardent. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 435–445, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P14-1041.

Shashi Narayan and Claire Gardent. Unsupervised sentence simplification using deep semantics. In *Proceedings of the 9th International Natural Language Generation conference*, pages 111–120, Edinburgh, UK, September 2016. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W16-6620.

Shashi Narayan, Claire Gardent, Shay B. Cohen, and Anastasia Shimorina. Split and rephrase. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 606–616, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1064.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Trans. Speech Lang. Process.*, 4(2), May 2007. ISSN 1550-4875. URL http://doi.acm.org/10.1145/1233912.1233913.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D Manning. LinGO redwoods: A rich and dynamic treebank for HPSG. *Research on Language and Computation*, 2(4):575–596, 2004.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P02-1040.

Terrence Parsons. Modifiers and quantifiers in natural language. *Canadian Journal of Philosophy, Supplementary Volume*, 6:29–60, 1980.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D14-1162.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N18-1202.

Carl Pollard and Ivan A. Sag. *Information-based syntax and semantics, Vol.1, Fundamentals.* Number 13 in CSLI lecture notes. Center for the Study of Language and Information, Stanford, CA, 1987.

Carl Pollard and Ivan A. Sag. *Head-driven phrase structure grammar*. Studies in contemporary linguistics. Center for the Study of Language and Information ; University of Chicago Press, Stanford : Chicago ; London, 1994.

Jean Pouget-Abadie, Dzmitry Bahdanau, Bart van Merriënboer, Kyunghyun Cho, and Yoshua Bengio. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 78–85, Doha, Qatar, October 2014. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W14-4009.

Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W09-1119.

Jonathon Read, Rebecca Dridan, Stephan Oepen, and Lars Jørgen Solberg. Sentence boundary detection: A long solved problem? In *Proceedings of COLING 2012: Posters*, pages 985–994, Mumbai, India, December 2012. The COLING 2012 Organizing Committee. URL https://www.aclweb.org/anthology/C12-2096.

Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1035.

Advaith Siddharthan. A survey of research on text simplification. *ITL - International Journal of Applied Linguistics*, 165:259–298, January 2014.

Jeff Speaks. Propositions are properties of everything and nothing. In Jeffrey C. King, Scott Soames, and Jeff Speaks, editors, *New Thinking About Propositions*, chapter 5, pages 71–90. Oxford University Press, Oxford, 2014.

Jana Straková, Milan Straka, and Jan Hajic. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-1527.

Stephanie Strassel, Jáchym Kolář, Zhiyi Song, Leila Barclay, and Meghan Glenn. Structural metadata annotation: Moving beyond English. In *6th INTERSPEECH*, volume 1, Lisboa, Portugal, September 2005. International Speech Communication Association.

Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Tsutomu Hirao, and Masaaki Nagata. Divide and translate: Improving long distance reordering in statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 418–427, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W10-1762.

Zoltán Gendler Szabó. Compositionality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition, 2017. URL https://plato.stanford.edu/archives/sum2017/entries/compositionality/.

Simone Teufel and Hans van Halteren. Evaluating information content by factoid analysis: Human annotation and stability. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 419–426, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-3254.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000. URL https://www.aclweb.org/anthology/W00-0726.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL https://www.aclweb.org/anthology/W03-0419.

Erik F. Tjong Kim Sang and Hervé Déjean. Introduction to the CoNLL-2001 shared task: clause identification. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*, 2001. URL https://www.aclweb.org/anthology/W01-0708.

Hans van Halteren and Simone Teufel. Examining the consensus between human summaries: Initial experiments with factoid analysis. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop*, volume 5, pages 57–64, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W03-0508.pdf.

Erik Velldal. *Empirical Realization Ranking*. PhD thesis, University of Oslo, Department of Informatics, 2008.

Zeno Vendler. Facts and events. In *Linguistics in Philosophy*, pages 12–146. Cornell University Press, Ithaca, New York, 1967.

Taro Watanabe, Eiichiro Sumita, and Hiroshi G. Okuno. Chunk-based statistical translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 303–310, Sapporo, Japan, July 2003. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P03-1039.

Michael White. Reining in CCG chart realization. In Anja Belz, Roger Evans, and Paul Piwek, editors, *Natural Language Generation. Lecture Notes in Computer Science*, volume 3123. Springer, Berlin, Heidelberg, 2004.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83, 1945.

Jie Yang and Yue Zhang. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-4013.

Jie Yang, Shuailong Liang, and Yue Zhang. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1327.

Gisle Ytrestøl. Technical Summary – Selection and Preprocessing of the WeScience Corpus. Technical report, Department of Informatics, Unviersity of Oslo, 2009.

Gisle Ytrestøl, Dan Flickinger, and Stephan Oepen. Extracting and Annotating Wikipedia Sub-Domains — towards a new eScience community resource. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7)*, pages 185–197, Groningen, The Netherlands, 2009.